HIERARCHICAL PATH PLANNING AND CONTROL OF A SMALL FIXED-WING UAV: THEORY AND EXPERIMENTAL VALIDATION

A Thesis
Presented to
The Academic Faculty

by

Dongwon Jung

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Aerospace Engineering

Georgia Institute of Technology
December 2007

HIERARCHICAL PATH PLANNING AND CONTROL OF A SMALL FIXED-WING UAV: THEORY AND EXPERIMENTAL VALIDATION

Approved by:

Professor Panagiotis Tsiotras,
Committee Chair
School of Aerospace Engineering
Georgia Institute of Technology

Professor Eric Feron
School of Aerospace Engineering
Georgia Institute of Technology

Professor Eric Johnson
School of Aerospace Engineering
Georgia Institute of Technology

Professor George Vachtsevanos
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Eric Corban
President
Guided Systems Technologies, Inc.

Date Approved: 30 October 2007

To my wife, whom I love,

and Elliot, my beloved son.

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Panagiotis Tsiotras for providing me with the opportunity to study and conduct research at one of the best engineering schools in the country. His constant guidance and encouragement have motivated me to reach my utmost accomplishment towards professional maturity. For all your support and concern for me and my family, I am deeply indebted to you for potitive influence you have had on my life. In addition, I thank the other Ph.D. committee members Dr. Eric Corban, Dr. Eric Feron, Dr. Eric Johnson, and Dr. George Vachtsevanos for taking time from their busy schedules to review my thesis and providing valuable suggestions to improve the quality of the dissertation. I especially extend my appreciation to Dr. Eric Corban for your kind efforts and sincere concern in regards to my future plans.

This research work has been supported in part by NSF (award no. CMS-0510259) and by the Georgia Tech President's Undergraduate Research Awards (PURA) program. Their finalcial support is greatly appreciated. I would like to acknowledge all of my lab-mates and alumni of the Dynamic and Control Systems Lab, Dr. Haijun Shen, Dr. Xiping Zhang, Dr. Hyungjoo Yoon, Ancil Marshall, Sachin Jain, Atri Dutta, Efstathios Bakolas, Raghu Cowlagi, Dae-Min Cho for their help and valuable discussions from I have benefitted tremendously. In particular, I would like to give special thanks to Dr. Byungmoon Kim who has led me towards the spacecraft experimental research in my early days of Georgia Tech. In addition, I thank to Dr. Brian Wilson and Dr. Efstathios Velenis who spent time together at the office, Montgomery Knight building Room 111, listening to my problems, successes, and offering advice. During the early phase of this research work, I would like to acknowledge the contributions of Emmanuel J. Levy, Debao Zhou, Rebecca Fink, Terry Williams, Jonathan

Moshe, Andrew Earl, Eiji Ozawa, and Jayant Ratti with whom I was able to carry on this research work. Also, I thank to Dr. Ravi Doraiswami, PRC, Georgia Tech for advising on the the PCB fabrication and Iven Cornery, Atlanta RC club for willingly assisting me during the flight tests as a remote pilot.

This acknowledgement is incomplete until I mention other colleagues and alumni in the control group of AE, Jeong Hur, Dr. Bong-Jun Yang, Dr. Nakwan Kim, Dr. Suresh Kannan, Dr. Yoonghyun Shin, Dr. Ali Kutay, Dr. Ramachandra Sattigeri, Jincheol Ha, Seung-Min Oh, Yoko Watanabe, Nimrod Rooz, Jongki Moon, Keumjin Lee, Allen Wu, Clauss Christmann, Jonathan Muse, Kilsoo Kim, Henrik B. Christophersen, Wayne J. Pickell, and Jeongjun Im, with whom I have enjoyed the time at Georgia Tech and have learned from interactions. I give special thanks to Dr. Bong-Jun Yang who gave valuable comments and suggesstions on my dissertation. At this point I would like to thank all of my Korean fellow students in AE who have supported and encouraged me morally during the years of Georgia Tech. I would like to recognize the efforts of Vivian O'Neal, Andrew Carignan and the other dedicated workers in AE, who provided administrative supports and research assistance.

I would like to express my deep gratitude with all my heart to my parents. Even from thousands of miles away, your endless support and encouragement have strengthened me to complete this long journey for Ph.D. I am also thankful to my parents-in-law and other family members who have always wished for my success. I don't think I could have made the journey without their constant support throughout my whole academic years. A special thank must be given to my beloved son Elliot Hyunje Jung who was born on October 29^{th}, 2007. Since he has become our new family member as an unborn child, he has been a great source of inspiration to me, making even the most stressful days brigher. Last but certainly not least, I would like to give thanks to my lovely wife, Eun Ju "Agatha" Song. Your embrace lends me courage and strength, your smile makes me happy, your passion inspires me, your devotion

spurs me to work hard, and your love fulfills me. Through you I have been centered in all respects of our upcoming life for a harmonious family. You are the best thing that has ever happened to me.

Dongwon "Thomas" Jung

November 13, 2007

TABLE OF CONTENTS

DEDICATION iii

ACKNOWLEDGEMENTS iv

LIST OF TABLES x

LIST OF FIGURES xi

LIST OF SYMBOLS OR ABBREVIATIONS xviii

SUMMARY xxii

I INTRODUCTION 1

1.1 Motivation 1

1.2 Literature Review 3

1.2.1 Geometric path planning 3

1.2.2 Obstacle avoidance path planning 4

1.2.3 Dynamic path planning 6

1.2.4 Multiresolution path planning 9

1.3 Outline of Thesis 10

1.4 Research Contributions 12

II MULTIRESOLUTION ON-LINE PATH PLANNING 15

2.1 Background 15

2.2 A Multiresolution Decomposition of $\mathcal{W}$ 16

2.2.1 The 2D wavelet transform 16

2.2.2 Fast lifting wavelet transform (FLWT) 18

2.2.3 Wavelet decomposition of the risk measure 21

2.3 Multiresolution Graph Connectivity 22

2.3.1 Computation of adjacency list from the FLWT 22

2.3.2 Cost assignment for $\mathcal{A}^*$ search 30

2.4 Multiresolution Path Planning 37

2.4.1 Multiresolution path planning algorithm 37
2.4.2 $\mathcal{D}^*$-lite path planning algorithm 39
2.5 Simulation Results 41
2.5.1 Simulation results for the proposed algorithm 41
2.5.2 Simulation results for the $\mathcal{D}^*$-lite algorithm 44
2.6 Comparison 44
2.7 Summary 47
III ON-LINE PATH SMOOTHING USING PATH TEMPLATES 50
3.1 Introduction 50
3.2 Tight Envelope for B-spline curves 52
3.2.1 Tight envelope for B-spline function 52
3.2.2 Tight envelope for planar B-spline curves 56
3.3 Obstacle avoidance path optimization 58
3.3.1 Channel constraints for obstacle avoidance 58
3.3.2 Smooth curve optimization 61
3.4 Construct path templates for different channels 63
3.4.1 Path rules within a finite horizon 64
3.4.2 Construct B-spline path templates 69
3.5 On-line path smoothing algorithm 70
3.5.1 Stitching the path segments 70
3.5.2 Simulation results of the on-line path smoothing algorithm . 77
3.6 Summary 79
IV PATH FOLLOWING CONTROL USING BACKSTEPPING AND PARAMETER ADAPTATION 81
4.1 Introduction 81
4.2 Problem Description 82
4.3 Path Following Controller Design 86
4.3.1 Kinematic controller design 86
4.3.2 Roll angle command via backstepping 89

4.3.3 Parameter adaptation 92
4.4 Simulation results 94
4.5 Summary 95
V REAL TIME IMPLEMENTATION OF THE HIERARCHICAL PATH CONTROL ALGORITHM USING HARDWARE-IN-THE-LOOP SIMULATION 103
5.1 Hardware description 103
5.2 Hardware-in-the-loop simulation environment 105
5.3 Simulation scenario 106
5.4 Simulation results 108
5.5 Summary 108
VI CONCLUSIONS AND FUTURE RESEARCH 115
6.1 Conclusions 115
6.2 Future Research 117
6.2.1 Reusable graph structure 117
6.2.2 Kinodynamically feasible trajectory generation using B-splines 118
6.2.3 Trajectory tracking controller design using differential flatness 119
6.2.4 Path planning in three dimensions 121
APPENDIX A UAV AVIONICS DESCRIPTION 123
APPENDIX B INERTIAL ATTITUDE AND POSITION REFERENCE SYSTEM DEVELOPMENTS 142
APPENDIX C MODELING AND HARDWARE-IN-THE-LOOP SIMULATION 163
APPENDIX D DESIGN OF INNER CONTROL LOOPS 192
REFERENCES 216
VITA 229

LIST OF TABLES

1 Computational cost of the proposed algorithm by the on-board autopilot. 45

2 The computational cost comparison between the multiresolution path planning v/s the $\mathcal{D}^*$-lite. 46

3 Number of self-avoiding walks on an $m \times n$ grid 67

4 Path templates for local path instances on the first quadrant. 69

5 Simulation parameters. 95

6 Specifications of the Rabbit RCM-3400 micro-controller module. . . . 123

7 Inertial sensors specifications 128

8 Specifications of the autopilot sensors. 129

9 Maximum deflection angles for each control surface. 141

10 Digital DATCOM input configuration file of the 1/5 scale Decathlon. 166

11 Static and dynamic stability derivatives and control derivatives estimated from the geometry of the 1/5 scale Decathlon. 167

12 Mass properties identified from experiments 168

13 Results of dynamic modeling of each control surface identified by experiments. 175

14 Body x-axis aerodynamic force coefficient (C_X) identification results. 183

15 Body z-axis aerodynamic force coefficient (C_Z) identification results. . 183

16 Body y-axis aerodynamic moment coefficient (C_m) identification results. 184

17 Body y-axis aerodynamic force coefficient (C_Y) identification results. 184

18 Body x-axis aerodynamic moment coefficient (C_ℓ) identification results. 185

19 Body z-axis aerodynamic moment coefficient (C_n) identification results. 185

LIST OF FIGURES

1 Thesis Outline. 11

2 A typical one-stage two-band filter banks used for implementing the discrete wavelet transform. 19

3 One step decomposition using the lifting scheme with the lazy wavelet. 20

4 Multiresolution representation of the environment according to the distance from the current location of the agent. 23

5 Multiresolution cell subdivision across different levels. 24

6 Recursive raster scan method for identifying independent cells. 25

7 Basic connectivity properties with respect to the location of the leaf cell. 27

8 Searching an adjacent cell along the left search direction. 28

9 Refined adjacency search algorithm. 29

10 Connectivity relationship constructed from the multiresolution cell decomposition over three levels. 29

11 Pseudo-code implementation of the adjacency search algorithm. . . . 31

12 Pseudo-code implementation of the adjacency search algorithm: Recursive link connection. 32

13 Pseudo-code implementation of the adjacency search algorithm: Recursive link connection (continued). 33

14 Computational cost for the adjacency search algorithm in terms of data size. 34

15 Computational cost for the adjacency search algorithm in terms of window size. 35

16 Pseudo-code implementation of proposed multiresolution path planning scheme. 38

17 Pseudo-code implementation of $\mathcal{D}^*$-lite path planning scheme. 42

18 Path evolution and replanning. Dashed-dot lines represent the currently tentative optimal path obtained from the $\mathcal{A}^*$ algorithm, based on the available multiresolution approximation of the environment at different time steps. Solid lines reveal the actual path followed by the agent. 48

19 Path evolution and replanning using the $\mathcal{D}^*$-lite algorithm. Dashed-dot lines show the currently tentative optimal path obtained from the $\mathcal{D}^*$-lite algorithm, based on the distance cost outside the high resolution area. The actual path followed by the agent is drawn by solid lines. . 49

20 B-spline basis functions N_j^3 over the knot $u \in [0,\ 1]$. 53

21 Non-negative and convex functions β_{ki}. 55

22 One dimensional cubic B-spline bounding envelopes. 56

23 Constructing the envelope of a planar curve from neighboring bounding boxes. 57

24 Bounding envelopes $\boldsymbol{e}_L$ and $\boldsymbol{e}_R$ of two-dimensional cubic B-spline . . . 58

25 Signed distance map for an arbitrary polygonal line. The feasible region is characterized by the negative function values. 60

26 Geometric constraints formulation. The channel is given by two polylines $\boldsymbol{\ell}_L$ and $\boldsymbol{\ell}_R$, the envelope of the B-spline is drawn by the dashed lines, which is supposed to stay inside the channel. 61

27 Two optimization results. 63

28 Examples of path sequences starting from the current cell at the center. We adopt the four-connectivity between cells. The goal cell is supposed to be located beyond the horizon. Possible path sequences are given as examples of the optimal path. The path A is written by `NENEN`$\cdots$, the path B is `EESE`$\cdots$, the path C is `SSEES`$\cdots$, and the path D is `WNNWW`$\cdots$. 65

29 Local path instances on the first quadrant. From the additional symmetry about the diagonal axis, it is possible to transform the path instance drawn in dashed line (`NEENE`) to the path instance drawn in solid line (`ENNEN`). Path rules are given in order to determine unique path instances reaching the cell at the top boundary. 66

30 Example incorporating the path templates on a complex path sequence. Five local path instances are are connected each other to reach the goal cell. The actual path words are equivalently recovered from the path templates with corresponding symmetry operations, which are horizontal(H), vertical(V), diagonal(D) reflections. 69

31 Path templates results from the path optimization using B-spline curves. Each plot corresponds to the local path instance in Table 4. 71

31 (Continued) Path templates results from the path optimization using B-spline curves. Each plot corresponds to the local path instance in Table 4. 72

31 (Continued) Path templates results from the path optimization using B-spline curves. Each plot corresponds to the local path instance in Table 4. 73

31 (Continued) Path templates results from the path optimization using B-spline curves. Each plot corresponds to the local path instance in Table 4. 74

32 Determine the unique $\boldsymbol{p}_2$ in terms of $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$ in conjunction with the design parameter R_a. 76

33 Example of stitching the two B-spline curve with a cubic B-spline curve 78

34 On-line path smoothing in conjunction with replanning using $\mathcal{D}^*$-lite algorithm. Dashed-dot lines represent the currently tentative optimal path obtained from the $\mathcal{D}^*$-lite algorithm, based on the distance cost outside the high resolution area. Actual path followed by the agent is derived from the B-spline path templates, which are represented by solid lines. 80

35 Definition of the Serret-Frenet frame for the path following problem. . 83

36 Local vector fields generated with respect to two distinct points on the path. 89

37 Reference path and actual trajectory of the UAV without parameter adaptation. 96

38 Error states without parameter adaptation. 97

39 Command inputs without parameter adaptation. 98

40 Reference path and actual trajectory of the UAV with parameter adaptation. 99

41 Error states with parameter adaptation. 100

42 Command inputs with parameter adaptation. 101

43 Parameter estimate of λ_ϕ. 102

44 Block diagram for control hierarchy of the proposed path control algorithm. 104

45 A small fixed-wing UAV equipped with an autopilot for hierarchical path planning control. 105

46 High fidelity hardware-in-the-loop simulation (HILS) environment that enables rapid testing of the proposed path planning algorithm. 106

47 Illustration of the on-line implementation of the proposed hierarchical path control algorithm. 107

48 Simulation results of the hierarchical path control implementation. Figures on the right show the close-up view of the simulation. At each instant the channel is drawn by polygonal lines, where the smooth path segment from the path templates stays. The actual path followed by the UAV is drawn on top of the reference path. 109

48 Simulation results of the hierarchical path control implementation. (cont'd) 110

48 Simulation results of the hierarchical path control implementation. (cont'd) 111

48 Simulation results of the hierarchical path control implementation. (cont'd) 112

48 Simulation results of the hierarchical path control implementation. (cont'd) 113

49 A 3D screen shot during the simulation. The ground track of the followed path is displayed showing that the UAV is avoiding the obstacles (for this case, it is the high elevation region). 114

50 System architecture of the UAV test-bed. 124

51 Sensor board design layout. The board is 5″ by 3″ printed circuit board (PCB). Four layers include the power plane and the ground plane (Not shown above). 125

52 Sensor board functional block diagram. 126

53 Assembled autopilot hardware. 127

54 Schematic diagrams of the designed autopilot: Power circuit and the micro-controller interface. 132

55 Schematic diagrams of the designed autopilot: The rate sensors and the accelerometers interface. 133

56 Schematic diagrams of the designed autopilot: Z-axis rate and acceleration, 3-axis magnetometer interface. 134

57 Schematic diagrams of the designed autopilot: Pressure sensors and the GPS interface. 135

58 Schematic diagrams of the designed autopilot: A-to-D converter and the servo switching interface. 136

59 The ground station GUI program. 137

60 Angular rate calibration results. 139

61 Accelerometer calibration results. 139

62 Effect of hard and soft iron disturbances and compensated magnetometer measurements. 140
63 Two different schemes for the implementation of the complementary filter. 143
64 Multiple measurements augmentation in the indirect complementary filter. 145
65 Entire complementary filter setup for pitch and heading angles. . . . 147
66 Kinematic relation at a truly banked turn condition. 147
67 System with a delayed measurement due to sensor latency. 151
68 GPS momentary outage during an aggressive maneuver. 153
69 Attitude estimation filters validation. 156
70 Navigation filter validation. 161
71 Geometric modeling of the 1/5 scale Decathlon used as an input to the DDATCOM program. 167
72 Torsional pendulum experimental setup for identifying the moment of inertia. 168
73 APC model airplane propeller 13″ by 8″. 170
74 Experimental setup for identifying static thrust force. 170
75 Rotating speed of propeller v/s the throttle command. 171
76 Static thrust comparison for a model propeller 13″ by 8″ 172
77 Thrust coefficient c_T v/s advance ratio for the selected propeller. . . . 173
78 Various doublet responses for dynamic modeling of the control surfaces. 174
79 A complementary filter in the feedback form. 175
80 Measured state variables during a longitudinal maneuver. 181
81 Measured state variables during a lateral maneuver. 182
82 Validation for the force coefficient C_X. 183
83 Validation for the force coefficient C_Z. 184
84 Validation for the moment coefficient C_m. 185
85 High fidelity hardware-in-the-loop (HIL) simulation environment. . . 186
86 Hardware-in-the-loop simulation screen shot 189

87 Root locus plot of the transfer function in Eq. (195) with a unity negative feedback of the pitch rate Δq with the positive gain k_q. As the gain increases, the short period mode gets poorly damped. 195

88 Root locus plot of the closed-loop system incorporating the lead compensator of the pitch rate. The closed-loop poles and zeros are shown, and the squares represent the location of the poles at $k_q = 0.101$. . . . 196

89 Root locus plot of the closed-loop system with the pitch angle PI controller as k_θ varies. The squares represent the closed-loop poles at $k_\theta = 0.855$. 197

90 Bode plot of the closed-loop system with the pitch angle PI controller. 198

91 Block diagram of the closed-loop pitch angle controller. The dashed line denotes the original PI controller implementation, whereas the dashed-dot line denotes an alternative PI implementation with closed-loop zero removed. 199

92 Step response of the closed-loop system by a unit pitch reference command. 199

93 Bode plot of the closed-loop system with the lead compensator. . . . 202

94 Root locus plot of the closed-loop system using a lead and PI compensators as k_h varies. The squares represent the closed-loop poles at $k_h = 1.22$. 203

95 Step response of the closed-loop system by a unit altitude reference command. 204

96 Integrator antiwindup technique with a single saturation nonlinearity. 204

97 Block diagram of the final closed-loop altitude controller. 205

98 Step response of the closed-loop system with the altitude controller, comparing the cases of with/without antiwindup scheme. 205

99 Root locus plot of the simplified speed control plant 206

100 Root locus plot of the closed-loop system incorporating the PI speed controller. The closed-loop poles and zeros are shown, and the squares represent the closed-loop poles at $k_{V_T} = 0.773$ 207

101 Step response of the closed-loop system by a unit speed reference command. 208

102 Root locus plot of the closed-loop system with the designed yaw washout damper. The squares represent the closed-loop poles at $k_r = 0.27$. . . 209

103 The effectiveness of the yaw damper to suppress the dutch roll mode oscillation. 210

104 Root locus plot of the closed-loop system with the roll damper. The squares represent the closed-loop poles at $k_p = 0.075$. 211

105 Root locus plot of the closed-loop system with the PI roll angle controller as k_ϕ varies. The squares represent the closed-loop poles at $k_\phi = 0.128$. 213

106 Step response of the closed-loop system by a unit roll angle reference command. 214

107 Bode plot of the closed-loop system with the roll angle PI controller. 214

108 Entire block diagram for the lateral controllers 215

LIST OF SYMBOLS OR ABBREVIATIONS

$\{u_k\}$	Non-decreasing knot sequence.
α	Angle of attack.
b	Span length.
$\bar{c}$	Aerodynamic reference chord length.
$\bar{L}$	Aerodynamic moment component about the body x-axis.
$\bar{L}_w$	Aerodynamic roll moment about the wind x-axis.
$\bar{q}$	Dynamic pressure.
β	Side slip angle.
$\boldsymbol{e}_L$	Left bounding envelope of a two dimensional B-spline curve.
$\boldsymbol{e}_R$	Right bounding envelope of a two dimensional B-spline curve.
$\boldsymbol{\ell}_L$	Two dimensional left channel polygon.
$\boldsymbol{\ell}_R$	Two dimensional right channel polygon.
C_D	Drag coefficient.
C_ℓ	Rolling moment coefficient.
χ	Course angle, inertial speed heading.
C_L	Lift coefficient.
C_m	Pitching moment coefficient.
C_n	Yawing moment coefficient.
C_y	Side force coefficient.
D	Aerodynamic drag force.
δ_a	Aileron deflection angle.
$\delta(\cdot)$	Approach angle.
δ_e	Elevator deflection angle.
δ_r	Rudder deflection angle.
e_d	Cross track error with respect to the path.

ℓ	Control polygon of B-spline.
ϵ_p	White noise process for roll rate gyro random walk process.
ϵ_r	White noise process for yaw rate gyro random walk process.
ϵ_w	White noise process for the wind speed bias random walk process.
e_s	Along track error with respect to the path.
η_E	Measurement noise of GPS position fix along the East direction.
η_h	Measurement noise of the altitude.
η_N	Measurement noise of GPS position fix along the North direction.
η_p	Roll rate gyro measurement noise.
η_r	Yaw rate gyro measurement noise.
η_w	Measurement noise of the airspeed.
FLWT	Fast lifting wavelet transform.
g	Earth's gravity constant.
GPS	Global Positioning System.
GUI	Graphical User Interface.
h	Altitude of the UAV.
HILS	Hardware in the loop simulation.
J	Advance ratio.
J_x	Moment of inertia of mass in the body x-axis.
J_{xz}	Product of inertia of mass in terms of the body x-y plane.
J_y	Moment of inertia of mass in the body y-axis.
J_z	Moment of inertia of mass in the body z-axis.
κ	Curvature of the curve.
L	Aerodynamic lift force.
λ_ϕ	Time constant of the first-order roll angle dynamics.
M	Aerodynamic moment component about the body y-axis.
$\mathcal{C}_d$	Multiresolution cell decomposition.

$\mathcal{F}$	Obstacle free configuration space.
$\mathcal{G}$	Topological graph structure.
$\mathcal{M}$	Collection of m integer distinct risk measure level.
$\mathcal{N}(\mathsf{x}, r)$	Neighborhood of the location x with distance r.
$\mathcal{O}$	Obstacle space.
$\mathcal{W}$	Two dimensional world enviroment.
$\mathsf{cell}_{\mathcal{G}}(v)$	Center of the cell correponding to the node v of the graph $\mathcal{G}$.
$\mathsf{node}_{\mathcal{G}}(\mathsf{x})$	Node of the graph $\mathcal{G}$ corresponds to the location x.
rm	Risk measure function at the location $\mathsf{x} = (x, y)$.
M_w	Aerodynamic pitch moment about the wind y-axis.
N	Aerodynamic moment component about the body z-axis.
N_j^d	jth B-spline basis function of degree d.
ν	Auxiliary control input for backstepping controller.
N_w	Aerodynamic yaw moment about the wind z-axis.
Ω	Rotational speed of propeller.
ω	Heading rate of the UAV.
ω_e	Error state for the heading rate.
p	Angular velocity component about the body x-axis.
p_b	Roll rate gyro bias.
p^D	Down position of the airplane in the NED frame.
p^E	East position of the airplane in the NED frame.
ϕ	Roll attitude angle.
p^N	North position of the airplane in the NED frame.
ψ	Heading angle.
q	Angular velocity component about the body y-axis.
r	Angular velocity component about the body z-axis.
r_b	Yaw rate gyro bias.

R_p	Propeller radius.
S	Aerodynamic reference area.
s	Arc length parameter.
θ	Pitch attitude angle.
$\mathbf{U}$	Velocity component in the body x-axis.
u_j^*	Greville abscissae.
V	Velocity component in the body y-axis.
V_T	Total airspeed in the wind x-axis.
V_w	Wind speed state.
W	Velocity component in the body z-axis.
$\widehat{\lambda}_\phi$	Estimate of the λ_ϕ.
$\widetilde{\chi}$	Error course angle.
X_A	Aerodynamic force component in the body x-axis.
X_T	Thrust force component in the body x-axis.
Y	Aerodynamic side force.
Y_A	Aerodynamic force component in the body y-axis.
Y_T	Thrust force component in the body y-axis.
Z_A	Aerodynamic force component in the body z-axis.
Z_T	Thrust force component in the body z-axis.

SUMMARY

Recently there has been a tremendous growth of research emphasizing control of unmanned aerial vehicles (UAVs) either in isolation or in teams. As a matter of fact, UAVs increasingly find their way into military and law enforcement applications (e.g., reconnaissance, remote delivery of urgent equipment/material, resource assessment, environmental monitoring, battlefield monitoring, ordnance delivery, etc.). This trend will continue in the future, as UAVs are poised to replace the human-in-the-loop during dangerous missions. Civilian applications of UAVs are also envisioned such as crop dusting, geological surveying, search and rescue operations, etc.

In this thesis we propose a new online multiresolution path planning algorithm for a small UAV with limited on-board computational resources. The proposed approach assumes that the UAV has detailed information of the environment and the obstacles only in its vicinity. Information about far-away obstacles is also available, albeit less accurately. The proposed algorithm uses the fast lifting wavelet transform (FLWT) to get a multiresolution cell decomposition of the environment, whose dimension is commensurate to the on-board computational resources. A topological graph representation of the multiresolution cell decomposition is constructed efficiently, directly from the approximation and detail wavelet coefficients. Dynamic path planning is sequentially executed for an optimal path using the $\mathcal{A}^*$ algorithm over the resulting graph. The proposed path planning algorithm is implemented on-line on a small autopilot. Comparisons with the standard $\mathcal{D}^*$-lite algorithm are also presented.

We also investigate the problem of generating a smooth, planar reference path from a discrete optimal path. Upon the optimal path being represented as a sequence

of cells in square geometry, we derive a smooth B-spline path that is constrained inside a channel that is induced by the geometry of the cells. To this end, a constrained optimization problem is formulated by setting up geometric linear constraints as well as boundary conditions. Subsequently, we construct B-spline path templates by solving a set of distinct optimization problems. For application in UAV motion planning, the path templates are incorporated to replace parts of the entire path by the smooth B-spline paths. Each path segment is stitched together while preserving continuity to obtain a final smooth reference path to be used for path following control.

The path following control for a small fixed-wing UAV to track the prescribed smooth reference path is also addressed. Assuming the UAV is equipped with an autopilot for low level control, we adopt a kinematic error model with respect to the moving Serret-Frenet frame attached to a path for tracking controller design. A kinematic path following control law that commands heading rate is presented. Backstepping is applied to derive the roll angle command by taking into account the approximate closed-loop roll dynamics. A parameter adaptation technique is employed to account for the inaccurate time constant of the closed-loop roll dynamics during actual implementation.

Finally, we implement the proposed hierarchical path control of a small UAV on the actual hardware platform, which is based on an 1/5 scale R/C model airframe (Decathlon) and the autopilot hardware and software. Based on the hardware-in-the-loop (HIL) simulation environment, the proposed hierarchical path control algorithm has been validated through on-line, real-time implementation on a small micro-controller. By a seamless integration of the control algorithms for path planning, path smoothing, and path following, it has been demonstrated that the UAV equipped with a small autopilot having limited computational resources manages to accomplish the path control objective to reach the goal while avoiding obstacles with minimal human intervention.

CHAPTER I

INTRODUCTION

1.1 Motivation

For decades, unmanned aerial vehicles (UAVs) have been considered for replacing human pilots during various missions. The applications of UAVs are increasing at a fast pace in military and in law enforcement missions (e.g., reconnaissance, remote delivery of urgent equipment/material, resource assessment, environmental monitoring, battlefield monitoring, ordnance delivery, etc [28, 113, 93]). Civilian applications of UAVs are also envisioned (crop dusting, geological surveying, search and rescue operations, etc). In particular, UAVs play an important role in replacing the human-in-the-loop during dangerous missions in hazardous or hostile environment.

A typical mission of a UAV is characterized by the requirement of navigating through or close to way points specified by the mission profile. The way points are usually specified a priori, and the mission must be completed satisfying certain criteria provided by the mission scenario. These criteria are minimum time, minimum fuel, minimum danger exposure, and so on. Assuming the domain of interest is fully characterized a priori, an off-line trajectory planning optimization can be used to obtain a satisfactory solution. However, in case of UAVs flying over hostile environment, where the domain is not known completely beforehand, UAVs necessarily require more advanced navigation and guidance capabilities [143, 13, 18]. Both trajectory design (planning) and trajectory tracking (control) tasks should be completely automated so that UAVs fly safely while avoiding obstacles and minimizing the exposure to danger. In addition, pop-up threats that might be encountered during missions need to be appropriately addressed during trajectory design and execution.

In line with the previous observations, it is natural to consider the trajectory planning problem as having two distinct objectives: The planned trajectory must eventually reach a final goal (after passing through the way points) skirting known obstacles, and should take into account any local environment variation, such as pop-up threats, by an online update of the planned trajectory. The former is considered as a global trajectory planning algorithm and the latter is as a local trajectory planning algorithm. The global trajectory planning determines overall performance of the generated trajectory whereas the local planning guarantees a feasible trajectory that can be followed by the UAV under dynamic constraints. The idea of combining both a global and a local algorithm seamlessly when designing a whole trajectory is similar to what a human pilot does. Given a final destination, the pilot plans the overall trajectory to be followed toward the destination avoiding known obstacles. Because distant information provided to the pilot is possibly insufficient to the accuracy than proximal information, he puts less emphasis on the distant information and utilizes more reliable local information to decide a feasible trajectory that keeps the airplane safe and flyable. In this work we plan to develop trajectory design and trajectory tracking algorithms that imitate 'human intelligence' in order to successfully complete complicated missions assigned to autonomous vehicles.

Assuming perfect knowledge of the environment while designing the trajectory, the global trajectory planner yields a better solution than the local planner and avoids local minima. However, the computational cost tends to increase as the problem size gets bigger. In contrast, a local planner can reduce the number of computation, but at the cost of a solution which often results in a locally entrapped trajectory. Thus, the combination of these two methods offers the best compromise between feasibility and optimality, while minimizing the computational cost so that they could be implemented in realtime.

1.2 Literature Review

1.2.1 Geometric path planning

Guidance, navigation, and control of mobile agents has been an important research topic for several decades. The applications are enormous in a variety of areas such as marine craft [108, 40], underwater vehicles [3, 74], unicycle type mobile robots [66], and unmanned aerial vehicles [102, 5]. The navigation control algorithms reported in the literature implement way-point following schemes assuming the vehicles are given a sequence of way-points a priori in the region of interest. These way points are connected in order to generate smooth path segments [61, 122] in a manner that they preserve the continuity of the curvature between line and arc segments, while minimizing the maximum curvature on the curve. A series of cubic splines was employed in Ref. [56] to connect the straight line segments in a near-optimal manner, whereas the algorithm presented in Ref. [6] yields extremal trajectories that transition between straight-line path segments smoothly in a time-optimal fashion.

In two-dimensional vehicle motion subject to non-holonomic constraints, such as a unicycle type mobile robot, it has been proved in Ref. [38] and [17] that for given initial and final configurations $p_0 = (x_0, y_0, \theta_0)$ and $p_f = (x_f, y_f, \theta_f)$, respectively, the optimal shortest path exists and is one among six types of paths: the paths consisting of at most three parts which are either straight line segments or arcs of circle of radius R if the vehicle has a maximum turning rate constraint. This set of path elements is termed as "Dubins path" and has inspired many researchers on how to design a feasible path by connecting way-points. Based on this idea, Bui and Souères [21] proposed new optimality conditions on these paths by computing a partition space from the two-dimensional plane of the configuration space, Yang and Kapila [151] exploited parameter optimization methods to solve optimal path planning problems. A similar geometry-based path planning algorithm is also reported in Ref. [80]. More recently, a three-dimensional extension of Dubins path was developed in Ref. [126].

Although the path planning strategies discussed above yield the optimal (shortest) paths between way-points for non-holonomic vehicles and yield easy implementation, one should note that they do not explicitly deal with obstacles possibly existing along the path segments. This disadvantage becomes apparent in the case when the vehicle navigates in an environment filled with obstacles where the designed path segments could collide with the obstacles. As discussed in Ref. [80] it is then necessary to consider a path planning algorithm that takes into account obstacles avoidance.

1.2.2 Obstacle avoidance path planning

One technique for obstacle avoidance is to use the "artificial potential field" concept. Khatib [65] presented the earlier result of real-time obstacle avoidance for mobile robots using this concept. In this method, a mobile robot moves in a potential field such that the goal is modeled as an attractive force and the obstacles and other vehicles are modeled as repelling forces. In Ref. [9], local minima of a potential field are connected in order to build an incremental graph, and the graph is searched to attain the final goal configuration. In Refs. [85, 14], an artificial potential field serves as a local holonomic planner to avoid obstacles, of which the output is modified in a least-squares sense to generate actual commands for the desired motion. Although potential field methods can produce smooth and dynamically feasible trajectories at low computational cost, they have several drawbacks. Detecting and avoiding the obstacles are mostly influenced by proximate obstacles, and the vehicle can get trapped in a local minimum. Another issue arises from the fact that obstacles are not modeled as hard constraints. The potential fields are typically modeled as continuous and differentiable functions, leading to an imprecise description of the obstacle's shape and dimension. This restriction could result in unsatisfactory obstacle avoidance when the vehicle maneuvers through tight environment.

In contrast to the local potential field techniques, the so-called global path planning algorithms, by which the overall workspace is accounted for during motion planning, can overcome not only the local minima problem but also guarantee tight obstacle avoidance. These algorithms begin with first obtaining a workspace representation explicitly incorporating the obstacles. A basic and simplest representation is to decompose the workspace into cells via a regular grid [141], but more efficient representations have been developed to reduce the memory cost for storing representations: Quadtrees [60] have been utilized for two dimensional spaces, and oct-tree [54] for three dimensional spaces. Roadmap type representations are utilized to extract a skeleton graph from the workspace, which keeps the connectivity between the nodes of feature points. Voronoi diagram [69] and Freeways [19] are widely used representations for global motion planning problem. The visibility graph [84, 101] is another method for obtaining a graph by connecting the edges (or, vertices) of obstacles from a given start and end points. Collin's decompositions [81] or probabilistic roadmaps [64] also become efficient representation methods for global path planning.

Having the workspace represented by either a partition of finite cells (region) or a complete graph, a path planning problem is then reduced to finding a sequence of neighboring cells or nodes in the graph satisfying certain extremal criteria. Several optimal searching algorithms have been proposed in the literature. The use of dynamic programming was proposed in Ref. [130] for a robot manipulator. An evolutionary search method was successfully used to compute near-optimal paths through obstacles and dynamically changing environment [115, 55]. The randomized rapid-exploring heuristic search in Ref. [63] forms a subset of stochastic optimization [91] for a global path planning, in which the search tree is expanded online towards randomly generated target states. If a graph is obtained for workspace representation, the $\mathcal{A}^*$ algorithm [49], or the Dijkstra's algorithm [35], are the most common algorithms to find the shortest path from a given initial node to a given goal node on the graph.

Many different versions of this search algorithm have been presented in the literature for path planning of mobile robots [7, 111, 112, 29, 68, 152].

A path planning algorithm over a dynamic environment has also been proposed in the literature. In order to deal with a dynamically changing environment, it is necessary for the path planning algorithm to replan as often as possible in accordance with the new information about the environment. Several replanning strategies have been proposed for locally-directed wandering [110], local modification of initial path [72], and obstacle perimeter detouring [86]. The $\mathcal{D}^*$ algorithm, proposed by Stentz [135, 136], is capable of planning a path in unknown or partially known environment, as it adopts an efficient incremental search scheme to produce an optimal path. The incremental search scheme incorporates the information from previous searches to find a solution much faster than solving each iteration from scratch. Subsequently, Koenig and Likhachev proposed the Lifelong Planning $\mathcal{A}^*$ (LPA) [70] which has been derived from $\mathcal{A}^*$ and adopted for dynamic environment by a reusing scheme. Furthermore, Koenig and Likhachev presented a $\mathcal{D}^*$-lite algorithm, derived from the LPA algorithm, which implements the same planning strategy as $\mathcal{D}^*$ but is algorithmically different.

1.2.3 Dynamic path planning

Most of the search methods discussed above do not incorporate dynamic models of the vehicles or robots. This, in turn, results in certain local path segments that the vehicles are hard to follow under kinematic constraints. Several authors have proposed to smooth these segments by taking the kinematic constraints into account [7, 29], but no explicit analysis on the optimality variation due to local path smoothing has been given.

Recently, mathematical optimization has been adopted by many researchers to

accommodate dynamic vehicle models in the path planning problems. Nonlinear programming was used in Refs. [114, 46] and the use of mixed integer linear programming (MILP) in the path planning problem was introduced in Refs. [124, 118, 117]. MILP allows inclusion of integer variables and discrete logic in a continuous linear optimization problem. These variables can be used to model logical constraints such as obstacle and collision avoidance, while dynamic equations of the vehicles are formulated as dynamic constraints. Although the MILP algorithm in trajectory planning can systemically handle *hard* obstacles of rectangular shapes over an entire workspace and provide a feasible trajectory if one exists, it can not handle the computational complexity for a large size trajectory planning problem, and the algorithm is not very useful in dynamically changing environment.

The computational complexity arising from the path planning over an entire workspace can be reduced by repeatedly solving a constrained optimization problem online over a finite planning horizon. This scheme is so called model predictive control (MPC), or receding horizon control (RHC), originated from the chemical process industry and has received great attention due to its broader applications over various research fields. A good survey on MPC/RHC can be found in Refs. [45, 97] and the relevant texts [24, 4]. In recent years, receding horizon control has been introduced to solve the problems of trajectory generation and collision avoidance of multiple vehicles: Fuller et al [43] presented a MPC-based trajectory generation method for a flight vehicle with nonlinear dynamics by using a feedforward nominal trajectory to pose a time-varying linear, convex optimization problem. This result is extended in Refs. [131, 77] to account for obstacle avoidance trajectory planning. Shim et al [67, 129] proposed a nonlinear model predictive tracking control (NMPTC) for planning a path under input and state constraints as well as collision avoidance. More recent publications regarding the implementation of NMPTC for UAVs are found in Refs. [128, 62]. The benefit of the MPC approach is that it can allow one

to effectively handle multi-variable systems with inputs and states constraints while less computational cost requires.

On the other hand, an alternative receding horizon approach based on MILP was introduced in Refs. [13, 119]. In this approach, a discrete point mass model was employed as a finer dynamics model for a two-dimensional vehicle under the velocity and force limits, which correspond to state and control constraints, respectively. A finer level of optimization was executed over a fine resolution of planning horizon using MILP solver, whereas a coarser level of optimization for the shortest path was done by searching a visibility graph that was expanded from a goal point through edges of each *hard* obstacle. These two-step optimizations achieve the shortest path over a long time-scale while planning a dynamically feasible path over a short time-scale at the reduced computational cost. In the continuation papers, the authors of Refs. [12, 73] proposed a stable trajectory for highly complex environment by putting turning circles at each obstacle's corner, which takes into account the kinematic constraints of the maximum turning rate of vehicles. In addition, a safety condition of the algorithm was imposed by the form of loiter circles as a backup plan in case of the algorithm failure [123]. Although this algorithm has been successfully tested and executed for UAV guidance [146, 125], the algorithm has certain underlying weaknesses. First, because the discrete MILP solver incorporates linearized equations, in conjunction with binary variables as the constraints of turning circles, velocity and force limits, and so on, a better accuracy of solution could be obtained from using more binary variables to accommodate detailed constraint equations. This results in increased computation time that may not be suitable for realtime implementation. Next, since the obstacles are limited to rectangular shapes to apply the visibility graph search, various shapes of obstacles other than rectangles will require a conservative modeling of obstacles by polygons that completely enclose such obstacles, which might end up with a conservative path.

1.2.4 Multiresolution path planning

The path planning methodology that will be developed in this thesis will involve a multiresolution workspace representation. One benefit of such representation is that it is able to handle complex shapes of obstacles other than rectangles [60]. In this hierarchical representation, the workspace is recursively divided into cells of either white (obstacle-free region) or black (obstacle region). Since the detail partitioning is done along the boundary of the obstacles, an arbitrary obstacle shape can be incorporated into the path planning problem. Finding the shortest path around the obstacles is obtained by searching over the hierarchical data structure containing the connectivity information between cells [103]. Extensions of this standard quadtree data structure were pursued in Refs. [30, 148] to overcome the limitation of generating a nonoptimal solution when an obstacle is located on or near the boundary of a quad. In addition, triangular cell decompositions [52] instead of a square cell of the quadtree are known to provide an alternative method to get around such limitation. Another benefit of employing multiresolution based path planning is that it can substantially reduce the complexity of computation by adopting variable resolution grids [11]. The resolution is set high close to a vehicle and becomes less at far away distance. Taking care of the connectivity between the non-uniform size neighboring cells, the path planner could find a high accuracy initial path and replan continuously as the vehicle moves at reduced computation costs.

Recently, a wavelet-based path planning algorithm has been introduced to efficiently solve the path planning problem on a complex environment. In Refs. [47, 121], a trajectory was first parameterized using wavelet decomposition and then optimized for satisfying mission requirements using dynamic optimization methods. Based on multiresolution active modeling, the planned trajectory was characterized as high fidelity over the near-term and as approximate over the longer-term. The wavelet transform can also provide a fast decomposition of the environment at different levels

of resolution [33] and it can be used to construct a hierarchical representation of the workspace as used in Ref. [104]. In a recent paper, Tsiotras and Bakolas [142] proposed to utilize the wavelet transform in a hierarchical on-line path planning problem, in which the workspace representation has high resolution close to the current position of the vehicle and low resolution far away. A topological graph is obtained from the wavelet decomposition online and repeatedly searched for the shortest path using Dijkstra's algorithm. The path planning algorithm is scalable, and can be implemented in real-time for various computational resources of the agent.

1.3 Outline of Thesis

Having presented the motivation and the related work in the literature, we now present the main objectives of this thesis. The outline of the overall organization of the thesis is shown in Fig. 1: In Chapter 2, we present the multiresolution path planning algorithm. We employ the fast lifting wavelet transform scheme to approximate a world by a multiresolution cell decomposition. An efficient algorithm for building the adjacency relationship is presented. In Chapter 3, the on-line path smoothing algorithm is presented. The path templates are comprised of a set of B-spline curves, which are computed from the off-line optimization. Given a discrete optimal path sequence, an on-line path smoothing algorithm is proposed to generate a smooth reference path to be used for path following control. Design of a nonlinear path following control law is presented in Chapter 4. We present a kinematic path following controller, which calculates the heading rate command input to the unicycle kinematics. The actual roll angle command for a fixed-wing UAV is derived from the heading rate command using backstepping technique, and the parameter adaptation scheme is incorporated to deal with the uncertainty of the time constant. In Chapter 5, hardware-in-the-loop (HIL) simulation results of the proposed multiresolution,

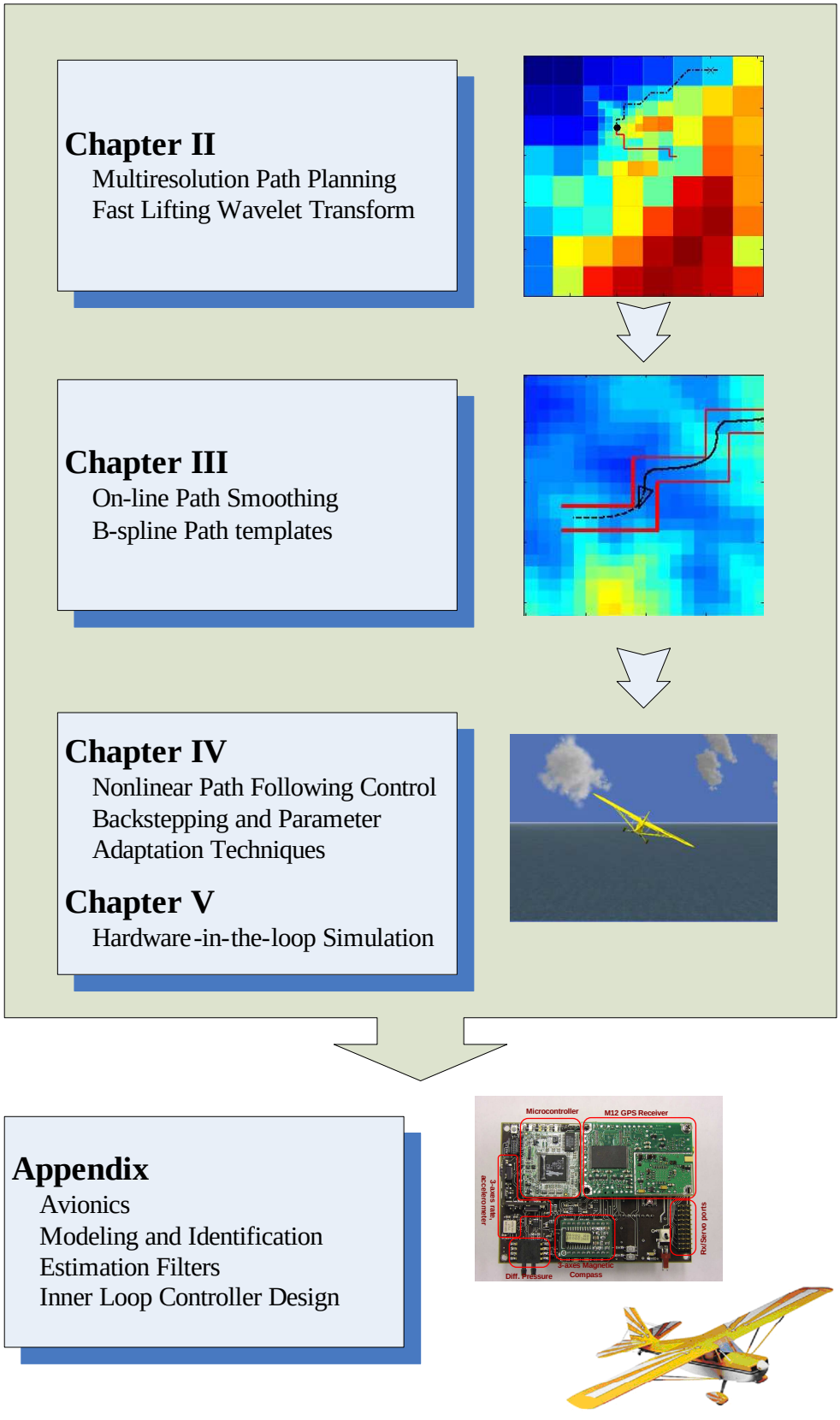


Figure 1: Thesis Outline.

hierarchical path control algorithm are shown. The proposed hierarchical path control algorithm is implemented on-board the actual hardware platform in real-time by a seamless integration of the hardware and the software. Finally, Chapter 6 summarizes the main results of this thesis, and enlists several recommendations for the future research.

1.4 Research Contributions

In this thesis we present both a theoretical development of the hierarchical path control algorithm and experimental results from the on-line implementation on the actual system. The following summary enlists the contribution of this work.

- Autonomous path planning for small UAVs imposes severe restrictions on control algorithm development, stemming from the limitations imposed by the on-board hardware and the requirement for on-line implementation. We propose a method to overcome this problem by using a new multiresolution path planning scheme. The fast lifting wavelet transform incorporating integer arithmetic is utilized to compute multiresolution representations of the environment, by taking into account the available computational resources of the agent. In addition, we show that the connectivity relationship of the graph $\mathcal{G}$ that is associated with the multiresolution representation can be constructed directly from the wavelet coefficients. Hence, the proposed multiresolution path planning algorithm is suitable for the on-line, on-board, and real-time implementation on a small UAV.

- For guidance and navigation control of UAVs, a complete solution which takes into account the equations of motion in conjunction with kinematic constraints is far from implementing in real-time, specially for small UAVs equipped with limited on-board hardware. We propose a path smoothing algorithm using a set of path templates. Instead of smoothing the entire path from an initial position

to the goal position, we smooth the path segments over a finite planning horizon with respect to the current position of the UAV. This approach is somewhat similar to the receding horizon control adapted to the path generation purpose. By incorporating the path templates from off-line optimization, the proposed smoothing scheme has the advantage of minimal on-line computational cost since most of computation is done off-line.

- We present a nonlinear path following control algorithm, to regulate the error distances from a reference path. Based on the kinematic control law for unicycle-type mobile robot in Refs. [75, 76], a backstepping control law is proposed to compute the actual roll angle command for a fixed-wing UAV. In addition, in order to compensate for the inaccurate time constant of the roll angle closed-loop system, we propose to apply the parameter adaptation technique. The performance of the proposed path following control law is validated through the realistic simulation environment, showing that the applicability of the proposed algorithm on the actual system.

- The proposed hierarchical path control algorithm, which includes the path planning at the top level hierarchy, the path smoothing in the middle, and the path following at the bottom level, is experimentally validated through a realistic hardware-in-the-loop simulation environment. We describe the practical issues associated with the implementation of the proposed control algorithm, taking into consideration the actual hardware limitation. With a seamless integration of the hardware and the software, we demonstrate a real-time validation of the proposed hierarchical path control algorithm on-board the autopilot.

- The heart of the UAV platform is its autopilot, which consists of a flight control computer, sensors, actuators, communication devices and peripherals, along with the associated software. In order to provide maximum flexibility in control

law development and implementation, we develop both the hardware and the low-level and high-level software in-house. We present the detail development of the autopilot hardware components and the subsystem integration process.

- A complete simulation environment for testing and validating UAV control systems is presented. With a comprehensive system modeling and experimental identification of a fixed-wing UAV, a high-fidelity simulation environment is built, which allows to incorporate a simulation-based testing during the development of the hardware and software. In addition, hardware-in-the-loop (HIL) simulation environment is developed for validating hardware and the software under realistic conditions. Details on developing user interface and subsystem modeling are also presented.

- A low cost inertial attitude and position reference system for a small UAV is presented. The estimation algorithms are simple, yet effective so that a microcontroller can execute these algorithms within a small time interval. We develop an algorithm that combines a complementary filter and a Kalman filter for the Euler attitude angles. A straightforward and innovative way of handling the data latency and the outage of a GPS sensor is introduced. Finally, a cascaded position estimation Kalman filter is designed utilizing the attitude estimates to lower the computational burden with a small performance loss.

CHAPTER II

MULTIRESOLUTION ON-LINE PATH PLANNING

2.1 Background

In a typical mission of a UAV, various sensors (e.g., cameras, radars, laser scanners, satellite imagery) having different range and resolution characteristics are employed to collect information about the environment the vehicle operates in. A computationally efficient path planning algorithm, specifically adopted for on-line implementation, should therefore choose the expedient information from all these sensors, and use the on-board computational resources to design the part of the path (spatial and temporal) that needs it most. In a nutshell, a computationally efficient algorithm suitable for *on-line* implementation should be characterized by a combination of short term tactics (reaction to unforeseen threats) with long-term strategy (planning towards the ultimate goal).

In this study, we assume a world environment $\mathcal{W} \subset \mathbb{R}^2$ that includes the obstacle space $\mathcal{O} \subset \mathcal{W}$ and the obstacle-free configuration space $\mathcal{F} = \mathcal{W} \setminus \mathcal{O}$ of all feasible states. We employ the wavelet transform to perform the required multiresolution decomposition of $\mathcal{W}$. The fast lifting wavelet transform (FLWT) offers a fast decomposition of a function at different levels of resolution, which can be twice as fast as the classical wavelet transform[1]. Furthermore, the FLWT integer arithmetic can be implemented to reduce the computational cost dramatically. This makes the FLWT especially suitable for processing data in a small micro-controller. The use of FLWT

[1]The computational complexity of the lifting scheme is still of order $\mathcal{O}(n)$ where n is the input data[145], however, the computational time may decrease by half according to wavelet basis.

has also the added benefit of allowing the construction of the associated cell connectivity relationship directly from the wavelet coefficients, thus eliminating the need for quadtree decomposition.

We employ the multiresolution path planning scheme to find an optimal path on the topological graph $\mathcal{G}$ induced by the multiresolution wavelet-based cell decomposition. Namely, the optimal path from an initial state to a final state may comprise of cells over different resolutions, depending on the distance from the current position of the agent. At the finer resolution level, the path is resolved through feasible states, hence avoiding obstacles. Multiresolution path planning is known to be more flexible, and computationally efficient for on-line implementation.

2.2 A Multiresolution Decomposition of $\mathcal{W}$

2.2.1 The 2D wavelet transform

The idea behind the wavelet transform is to represent a function $f \in \mathcal{L}^2(\mathbb{R})$ via a linear combination of elementary basis functions $\phi_{J,k}$ and $\psi_{j,k}$ as follows

$$f(x) = \sum_{k\in\mathbb{Z}} a_{J,k}\phi_{J,k}(x) + \sum_{j\geq J}\sum_{k\in\mathbb{Z}} d_{j,k}\psi_{j,k}(x), \tag{1}$$

where $\phi_{J,k}(x) = 2^{J/2}\phi(2^J x - k)$ and $\psi_{j,k} = 2^{j/2}\psi(2^j x - k)$. The choice of J determines the low resolution, or coarse approximation of f, spanned by the scaling function $\phi_{J,k}(x)$. The rest of $\mathcal{L}^2(\mathbb{R})$ is spanned by the wavelet functions $\psi_{j,k}(x)$ which provide the higher, or finer resolution details of the function. In other words, when analyzing the function f at the coarsest level (low resolution), only the most salient features of f will be revealed. Adding finer levels (high resolution) implies adding more and more details of the function f. The expansion (1) thus reveals the properties of f at different levels of resolution[22, 90]. In addition, in the ideal case both the scaling function and the wavelet function have compact support, i.e., they are non-zero only on a finite interval. This allows the wavelets to capture the localized features of the function f.

The wavelet transform can be readily extended to the two-dimensional case by introducing the following families of functions

$$\Phi_{j,k,\ell}(x,y) = \phi_{j,k}(x)\phi_{j,\ell}(y), \tag{2a}$$

$$\Psi^1_{j,k,\ell}(x,y) = \phi_{j,k}(x)\psi_{j,\ell}(y), \tag{2b}$$

$$\Psi^2_{j,k,\ell}(x,y) = \psi_{j,k}(x)\phi_{j,\ell}(y), \tag{2c}$$

$$\Psi^3_{j,k,\ell}(x,y) = \psi_{j,k}(x)\psi_{j,\ell}(y). \tag{2d}$$

Given a function $f \in \mathcal{L}^2(\mathbb{R}^2)$ we can then write

$$f(x,y) = \sum_{k,\ell\in\mathbb{Z}} a_{J,k,\ell}\Phi_{J,k,\ell}(x,y) + \sum_{i=1}^{3}\sum_{j\geq J}\sum_{k,\ell\in\mathbb{Z}} d^i_{j,k,\ell}\Psi^i_{j,k,\ell}(x,y) \tag{3}$$

where, for the case of orthonormal wavelets, the approximation coefficients are given by[2]

$$a_{j,k,\ell} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\,\Phi_{j,k,\ell}(x,y)\,\mathrm{d}x\,\mathrm{d}y, \tag{4}$$

and the detail coefficients by

$$d^i_{j,k,\ell} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\,\Psi^i_{j,k,\ell}(x,y)\,\mathrm{d}x\,\mathrm{d}y. \tag{5}$$

The key property of wavelets used in this paper is the fact that the expansion (3) induces the following multiresolution decomposition of $\mathcal{L}^2(\mathbb{R}^2)$

$$\mathcal{L}^2(\mathbb{R}^2) = \mathcal{V}_J \oplus \mathcal{W}_J \oplus \mathcal{W}_{J+1} \oplus \cdots \tag{6}$$

where $\mathcal{V}_J = \overline{\mathrm{span}_{k,\ell\in\mathbb{Z}}\{\Phi_{J,k,\ell}\}}$ and $\mathcal{W}_j = \overline{\mathrm{span}_{k,\ell\in\mathbb{Z}}\{\Psi^1_{j,k,\ell}, \Psi^2_{j,k,\ell}, \Psi^3_{j,k,\ell}\}}$ for $j \geq J$.

In this study we use the Haar wavelet system for reasons that will become apparent shortly. The Haar scaling function

$$\phi(x) = \begin{cases} 1 & \text{if } x \in [0,1), \\ 0 & \text{otherwise,} \end{cases} \tag{7}$$

[2]In the more general case of biorthogonal wavelets, projections on the space spanned by the dual wavelets and dual scaling functions should be used in (4) and (5).

and the Haar wavelet function

$$\psi(x) = \begin{cases} 1 & \text{if } x \in [0, 1/2), \\ -1 & \text{if } x \in [1/2, 1), \\ 0 & \text{otherwise,} \end{cases} \tag{8}$$

have compact support on $[0, 1]$. Hence, each scaling function $\phi_{j,k}(x)$ and wavelet function $\psi_{j,k}(x)$ in the Haar system has support on the dyadic interval $I_{j,k} \triangleq [k/2^j, (k+1)/2^j]$ of length $1/2^j$ and does not vanish in this interval [22, 149]. Similarly, we may associate the two-dimensional scaling function $\Phi_{j,k,\ell}$ and the wavelet function $\Psi^i_{j,k,\ell}$ $(i = 1, 2, 3)$ with the rectangular cell $c^j_{k,\ell} \triangleq I_{j,k} \times I_{j,\ell}$.

2.2.2 Fast lifting wavelet transform (FLWT)

Implementing the wavelet transform in practice requires dealing with a discrete signal. The basic step in a typical discrete wavelet transform (DWT) involves the use of filter banks. Figure 2 shows a discrete signal a_n filtered by two complementary high- and low-pass (decomposition) filters $\bar{g}$ and $\bar{h}$ before it is down-sampled. The results of this operation are the next coarser approximation and detail coefficients a_{n-1} and d_{n-1}, each containing half as many samples as the input signal a_n. For the inverse transform, first the signals a_{n-1} and d_{n-1} are upsampled by inserting zeroes between every sample. Subsequently, the two signals are filtered by the low- and high-pass (reconstruction) filters $\tilde{g}$ and $\tilde{h}$, respectively, and then added together. This sequence of operations results in perfect reconstruction of the original signal a_n. Details of the filter bank implementation of wavelet transforms can be found, for instance, in Refs.[37] and [138].

The fast lifting wavelet scheme, originally introduced in Refs. [139] and [140], is a new method for building wavelets directly in the time domain, thus avoiding the use of Fourier analysis. Moreover, the scheme can be extended to construct the so-called second generation wavelets, which have certain benefits for handling boundary effects,

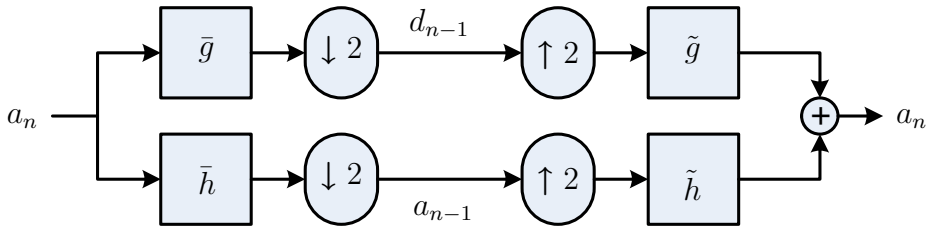


Figure 2: A typical one-stage two-band filter banks used for implementing the discrete wavelet transform.

irregular samples, and arbitrary weight functions[138].

A typical lifting decomposition scheme is depicted in Fig. 3. The first block in this decomposition splits the original signal a_n into two disjoint sets of samples containing the odd and the even indexed samples (Lazy wavelet). Because the even and odd subsets are correlated to each other locally, each signal is lifted by the opposite signal after passing through the corresponding operators $\mathcal{P}$ and $\mathcal{U}$ (the dual and primal lifting, or prediction and update, respectively). Finally, the results are normalized with the constants k_a and k_d, to end up with the approximation and detail coefficients, a_{n-1} and d_{n-1}, respectively.

For the case of the unnormalized Haar transform, the dual lifting does nothing more but calculate the difference

$$d_{n-1,k} = a_{n,2k+1} - a_{n,2k}, \tag{9}$$

whereas the primal lifting calculates the coarser approximation coefficients having the same average value as the original signal, by updating the even samples using the previously calculated detail signal as follows

$$a_{n-1,k} = a_{n,2k} + d_{n-1,k}/2. \tag{10}$$

It has been proved that all classical wavelet transforms can be implemented using the lifting scheme[33]. Most interestingly, the inverse transform is readily found by reversing the order of the operations and by flipping the signs.

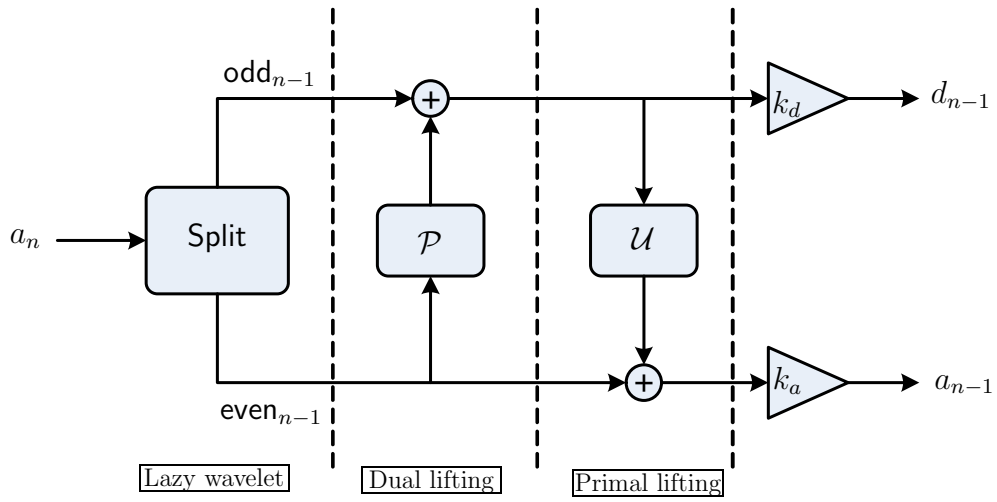


Figure 3: One step decomposition using the lifting scheme with the lazy wavelet.

The lifting scheme has a number of algorithmic advantages, such as faster computation speed (twice as fast as the usual discrete wavelet transform), *in-place* calculation of the coefficients (that saves memory), immediate inverse transform, generality for extension to irregular problems, etc. In particular, the lifting scheme is applicable to many applications where the input data consists of integer samples. Unlike the typical wavelet transform where floating number arithmetic is implicitly assumed, the lifting scheme can be easily modified to map integers to integers, and is readily reversible to allow perfect reconstruction[23]. This reconstruction is possible by adopting the sequential transform after modifying Eqs. (9) and (10) as follows[50]

$$\begin{aligned} d_{n-1,k} &= a_{n,2k+1} - a_{n,2k}, \\ a_{n-1,k} &= a_{n,2k} + \lfloor d_{n-1,k}/2 \rfloor, \end{aligned} \tag{11}$$

where, $\lfloor \cdot \rfloor$ is the rounding operator. In the sequel, we use the fast lifting Haar transform for two-dimensional signals of integer samples using a sequential 2D scheme; that is, we perform two successive one-dimensional transforms through the rows and then columns of the input data.

2.2.3 Wavelet decomposition of the risk measure

Without loss of generality, we let $\mathcal{W} = [0,1]\times[0,1]$, which is described using a discrete (fine) grid of $2^N \times 2^N$ dyadic points. The finest level of resolution $J_{\max}$ is therefore bounded by N. It follows from Eq. (3) and the accompanying discussions that the Haar wavelet decomposition at resolution level $J \geq J_{\min}$ is given by

$$f(x,y) = \sum_{k,\ell=0}^{2^J-1} a_{J,k,\ell}\Phi_{J,k,\ell}(x,y) + \sum_{i=1}^{3}\sum_{j=J}^{N-1}\sum_{k,\ell=0}^{2^j-1} d^i_{j,k,\ell}\Psi^i_{j,k,\ell}(x,y), \tag{12}$$

and it induces a cell decomposition of $\mathcal{W}$ of square cells of maximum size $1/2^J \times 1/2^J$.

Assume now that we are given a function $\mathsf{rm} : \mathcal{W} \mapsto \mathcal{M}$ that represents the "risk measure" at the location $\mathsf{x} = (x,y)$, where $\mathcal{M}$ is a collection of m integer distinct risk measure levels defined by

$$\mathcal{M} \triangleq \{M_i : M_1 < M_2 < \cdots < M_m\}. \tag{13}$$

The obstacle space $\mathcal{O}$ is defined as the space where the risk measure values exceed a certain threshold $\overline{M}$, that is,

$$\mathcal{O} = \{\mathsf{x} \in \mathcal{W} \mid \mathsf{rm}(\mathsf{x}) > \overline{M},\ \overline{M} \in \mathcal{M}\}. \tag{14}$$

For $\mathsf{x} \in \mathcal{F}$, we may think of $\mathsf{rm}(\mathsf{x})$ as an indication of the proximity of the agent to the obstacle space, or the probability of $\mathsf{x} \in \mathcal{O}$.

We construct approximation of $\mathcal{W}$ at distinct levels of resolution $J_{\min} \leq j \leq J_{\max}$ at ranges r_j from the current location of the agent $\mathsf{x}_0 = (x_0, y_0)$, in the sense that the resolution j is used for all points inside the neighborhood

$$\mathcal{N}(\mathsf{x}_0, r_j) \triangleq \{\mathsf{x} \in \mathcal{W} : \|\mathsf{x} - \mathsf{x}_0\|_\infty \leq r_j\}. \tag{15}$$

where $r_{J_{\max}} \leq r_j \leq r_{J_{\min}}$. By this, we imply that the finer resolution $J_{\max}$ is used for points close to the current location, and coarser resolutions at different levels are used elsewhere, according to the distance from the current location. Hence, the

representation of $\mathcal{W}$ gets coarser farther away from the current location. Figure 4 illustrates this situation. The choice of $J_{\max}$ is determined by the requirement that at this level all cells can be resolved into either free or obstacle cells. The choice of $J_{\min}$ as well as the window span r_j are dictated by the on-board computational resources.

Let now $\mathcal{I}(j) \triangleq \{0, 1, 3, \cdots, 2^j - 1\}$ and let

$$\mathcal{K}(j) \triangleq \{k \in \mathcal{I}(j) \mid I_{j,k} \cap [x_0 - r_j, x_0 + r_j] \neq \varnothing\}, \tag{16a}$$

$$\mathcal{L}(j) \triangleq \{\ell \in \mathcal{I}(j) \mid I_{j,\ell} \cap [y_0 - r_j, y_0 + r_j] \neq \varnothing\}. \tag{16b}$$

Then the wavelet decomposition of rm, given by

$$\mathsf{rm}(x,y) = \sum_{k,\ell \in \mathcal{I}(J_{\min})} a_{J_{\min},k,\ell} \Phi_{J_{\min},k,\ell}(x,y) + \sum_{i=1}^{3} \sum_{j=J_{\min}}^{J_{\max}-1} \sum_{\substack{k \in \mathcal{K}(j) \\ \ell \in \mathcal{L}(j)}} d^i_{j,k,\ell} \Psi^i_{j,k,\ell}(x,y), \tag{17}$$

induces, with a slight abuse of notation, the following multiresolution cell decomposition on $\mathcal{W}$

$$\mathcal{C}_d = \Delta C_d^{J_{\min}} \oplus \cdots \oplus \Delta C_d^{J_{\max}}, \tag{18}$$

where, ΔC_d^j is a union of cells $c^j_{k,\ell}$ of dimension $1/2^j \times 1/2^j$.

2.3 Multiresolution Graph Connectivity

2.3.1 Computation of adjacency list from the FLWT

In the previous section we described the construction of a multiresolution cell decomposition $\mathcal{C}_d$ of $\mathcal{W}$ in Eq. (18) using the FLWT. We now assign a topological graph $\mathcal{G} = (V, E)$ to $\mathcal{C}_d$ as follows. The nodes which belong to the set V represent the cells $c^j_{k,\ell}$ in $\mathcal{C}_d$ and the edges in the set E represent the connectivity relationship between these nodes. In this section we show that the connectivity of the graph $\mathcal{G}$ can be constructed directly from the wavelet coefficients. Equivalently, we compute the adjacency list of $\mathcal{G}$ directly from wavelet coefficients obtained from the FLWT.

Since the scaling function $\Phi_{j,k,\ell}$ and the wavelet functions $\Psi^i_{j,k,\ell}$ $(i = 1, 2, 3)$ of the 2D Haar wavelet are associated with square cells $c^j_{k,\ell}$, the corresponding approximation and nonzero detail coefficients encode the necessary information regarding the

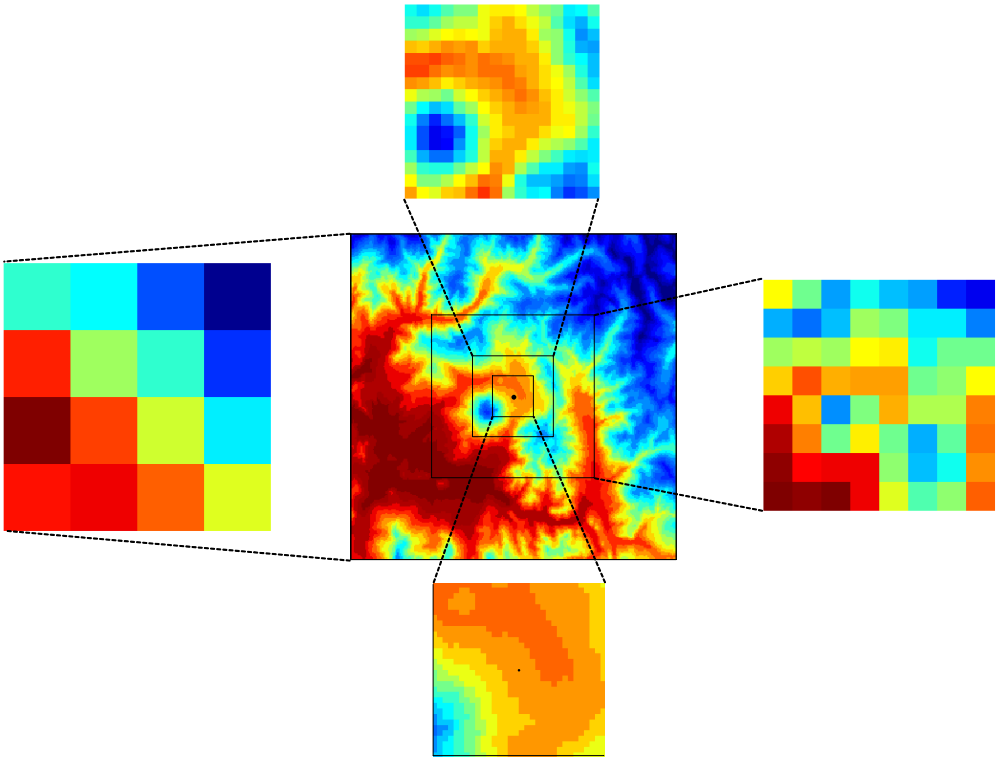


Figure 4: Multiresolution representation of the environment according to the distance from the current location of the agent.

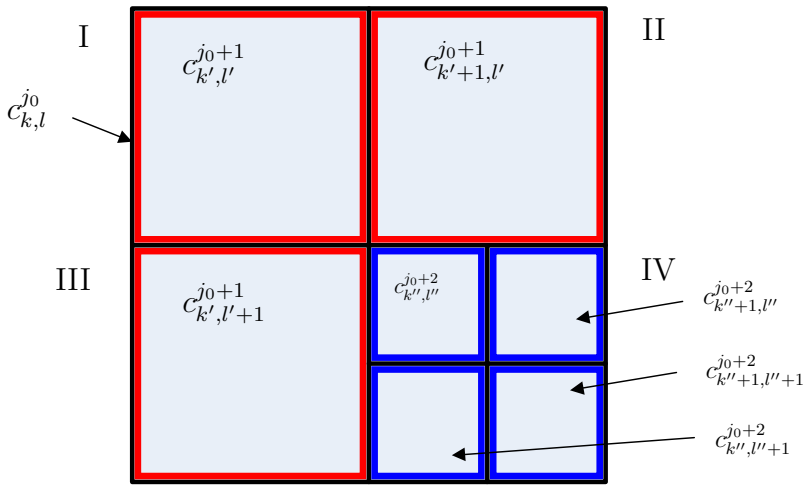


Figure 5: Multiresolution cell subdivision across different levels.

cell geometry (size and location). Recall that the approximation coefficients are the average values of the risk measure over the cells, and the detail coefficients determine the size of each cell. More specifically, consider a cell $c^{j_0}_{k,\ell}$ at level j_0, whose dimension is $1/2^{j_0} \times 1/2^{j_0}$ and is located at (k, ℓ). A cell will be called *independent* if it is associated with a non-zero approximation coefficient $a_{j_0,k,\ell}$, while the corresponding detail coefficients $d^i_{j,k,\ell}$ $(i = 1, 2, 3)$ at level $j_0 \leq j \leq J_{\max}$ are all zero. Otherwise, the cell is marked as a *parent* cell, and is subdivided into four *leaf* cells at level $j_0 + 1$. If a leaf cell cannot be subdivided further, it is classified as an independent cell. In Fig. 5, the top-most parent cell $c^{j_0}_{k,\ell}$ is subdivided into three independent cells at level $j_0 + 1$ with each non-zero approximation coefficient in the quadrant I, II, and III (all zero detail coefficients). For quadrant IV, the cell is further subdivided into four independent leaf cells at level $j_0 + 2$.

Assume that we are given the Haar wavelet transform of the risk measure function rm up to the level $J_{\min}$. The coarsest level of the cell dimension is set to $J_{\min}$. In Fig. 6, the initial coarse grid is drawn on the left. The agent is located at $\mathsf{x} = (x, y)$ and the high resolution horizon is given by r. Recalling expressions (16), we distinguish cells at distinct resolution levels, by starting from a coarse cell $c^{j_0}_{k,\ell}$, and by determining

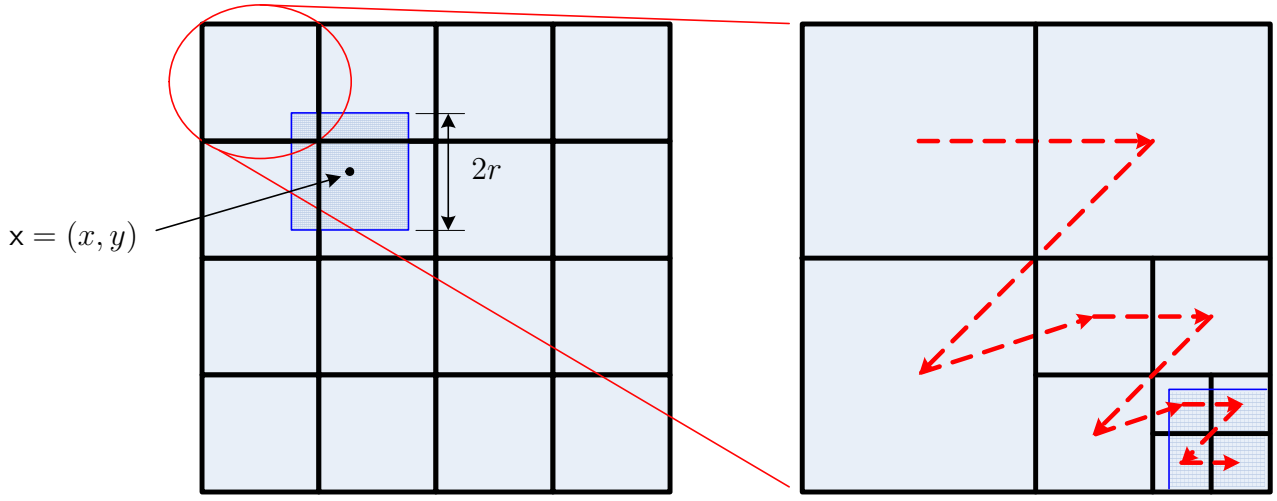


Figure 6: Recursive raster scan method for identifying independent cells.

if the cell either partially intersects or totally belongs to the set $\mathcal{N}(\mathsf{x}, r)$. The cell $c_{k,\ell}^{j_0}$ is easily ascertained to satisfy this property by choosing the indices such that $(k, \ell) \in (\mathcal{K}(j_0), \mathcal{L}(j_0))$. If the cell needs to be subdivided into higher resolution cells, the inverse fast lifting wavelet transform is first performed on the current cell (local reconstruction) in order to recover the four approximation coefficients at level $j_0 + 1$ and the corresponding detail coefficients. We then adopt the raster scan method[95] (zigzag search: I→II→III→IV) to examine each cell inside the parent cell overlapping with $\mathcal{N}(\mathsf{x}, r)$. This procedure is recursively repeated until the maximum resolution level $J_{\max}$ is reached. Figure 6 illustrates the recursive raster scan search. Once a cell is recognized as independent, we assign a node in the graph $\mathcal{G}$ with the node cost being the approximation coefficient that represents the average risk measure over the cell. In addition, the detail coefficients associated with the current cell are all set to zero; this will provide the necessary connectivity information between the cells later on.

After a cell has been identified as an independent cell, we search the adjacent cells in order to establish the adjacency relationship with the current cell. Recall that two

cells c_i and c_j are adjacent if

$$\partial c_i \cap \partial c_j \neq \varnothing, \quad i \neq j,$$

where ∂c_i denotes the boundary of the cell c_i. For our case of square cells, this implies that two cells are adjacent only along the following eight directions: Left, top, right, bottom, and the four diagonal directions. Following the recursive raster search for cell identification, the adjacency search requires establishing links between two cells that have been identified as independent cells. Recalling that the raster search progresses from left to right and from top to down (zigzag progress) as illustrated in Fig. 6, we confine the adjacency search to the following directions: Left, top-left, top, and top-right from the current cell. By doing this, we render half of the links (for eight-connectivity) to be connected from the current cell, and the remaining links with the current cell will be connected as the recursive raster scan progresses to the next cells. In addition, because we deal with cells of different dimensions, it is required to devise a generic method to find the adjacency relationship between the cells.

Figure 7 illustrates the basic search direction of each leaf cell inside a parent cell. The dashed arrow points towards an external search region, that is, an adjacent cell could be found beyond the parent cell, whereas the solid arrow points towards an internal search region that belongs to the parent cell. In each search, we implicitly assume that the level of adjacent cells may vary from that of the parent cell to $J_{\max}$ (external connection), or from that of the current cell to $J_{\max}$ (internal connection).

A leaf cell inherits the search region from its parent cells, whose search direction ends up with one of the solid arrows in Fig. 7. Figure 8 shows this inheritance property. In Fig. 8 the current cell is chosen to be $c_{\mathrm{I}}^{j_0+2}$. This cell is a leaf cell of the parent cell $c_{\mathrm{IV}}^{j_0+1}$, which further becomes a leaf cell of the top-most parent cell $c_{k,l}^{j_0}$. The cell $c_{\mathrm{IV}}^{j_0+1}$ is located on the fourth quadrant inside the top-most parent cell $c_{k,l}^{j_0}$ so that the search region for $c_{\mathrm{IV}}^{j_0+1}$ ends up with the internal searches at the level j_0+1, whose adjacency search property is inherited to the cell $c_{\mathrm{I}}^{j_0+2}$ for left, top-left, and

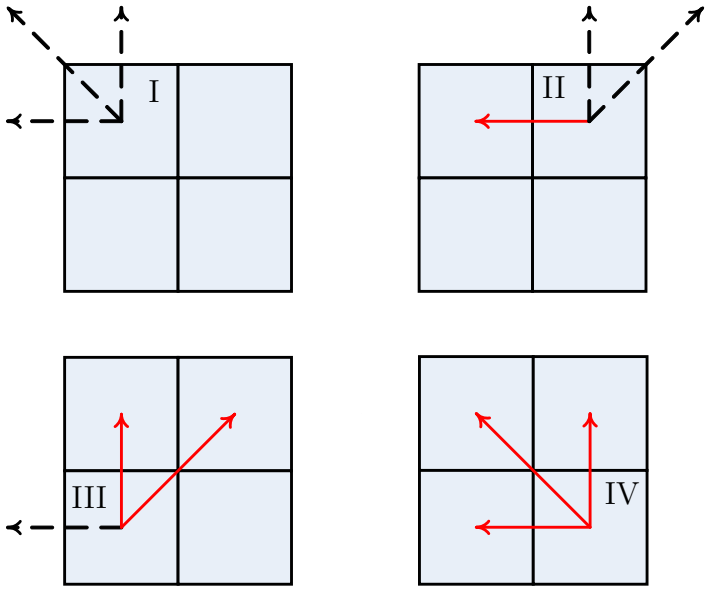


Figure 7: Basic connectivity properties with respect to the location of the leaf cell.

top direction searches. Having ascertained the basic search directions, we refine the adjacent search looking for opposite cells which must be independent and adjacent to the current cell. Because the opposite cells of the current cell could have different dimensions, we establish links by examining the associated detail coefficients of the opposite cells. Along the left search direction of $c_{\mathrm{I}}^{j_0+2}$, as illustrated in Fig. 8, one finds that only one independent cell at level j_0+1 is linked to $c_{\mathrm{I}}^{j_0+2}$.

The adjacency search algorithm refines its search to the higher levels if the opposite cell is not an independent cell, that is, if it is comprised of finer cells. This refinement subsequently forces a search of cells of the finer dimension (level) which are neighboring to the current cell. Subsequently, the detail coefficients of the opposite cells are examined in order to find the next finer cell that is adjacent to the current cell. For the top-left search direction of $c_{\mathrm{I}}^{j_0+2}$, as illustrated in Fig. 9(a), the search process initially examines the cell $c_{\mathrm{I}}^{j_0+1}$ located at the top-left corner of the current cell through the corresponding detail coefficient. Provided that the detail coefficient associated with the cell $c_{\mathrm{I}}^{j_0+1}$ takes a non-zero value, the cell is not an independent cell. Subsequently, the cell $c_{\mathrm{I}}^{j_0+1}$ is subdivided and the search process repeats at level

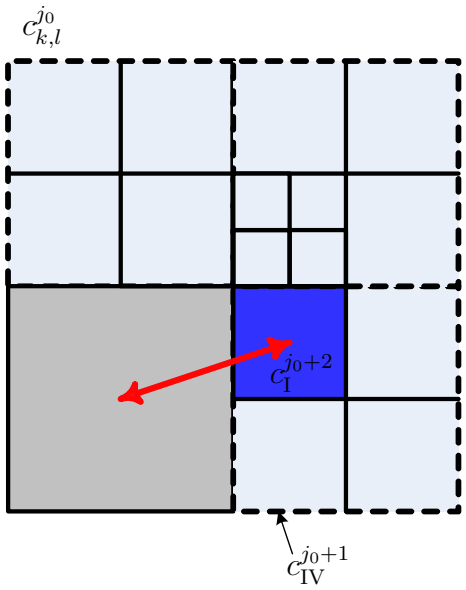
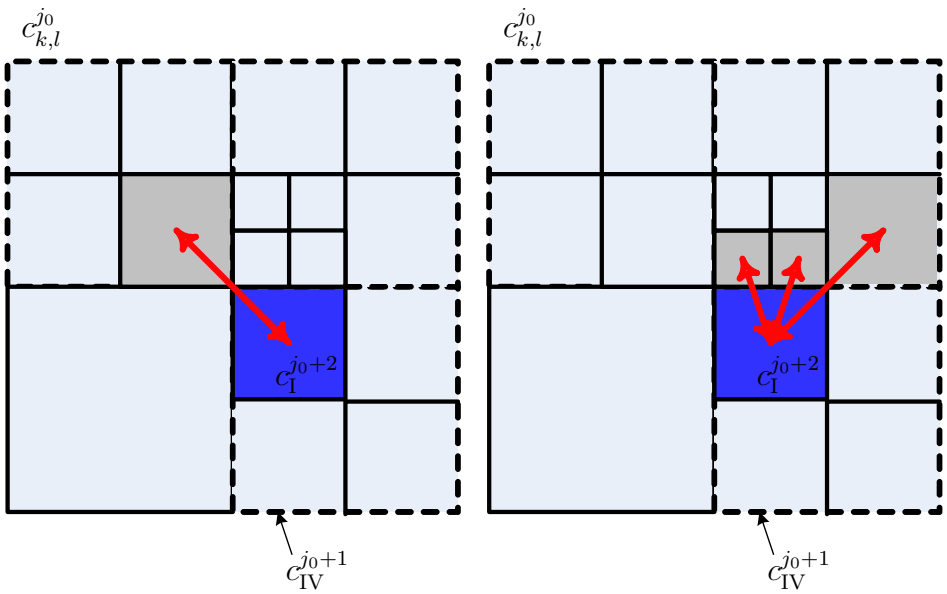


Figure 8: Searching an adjacent cell along the left search direction.

j_0+2 when the opposite cell to the current cell becomes an independent adjacent cell. In Fig. 9(a), since there exists no other independent cells along the top-left direction except the shaded one, a bidirectional link is established between the current and the opposite cells.

Similarly, for the top search direction, two cells at level j_0+3 and one at level j_0+2 are found to be independent and adjacent to the current cell. The bidirectional links are accordingly connected from the current cell $c_{\mathrm{I}}^{j_0+2}$ to those adjacent cells. Figure 9(b) depicts this situation. Finally, Fig. 10 shows an example of the graph structure obtained from the multiresolution cell decomposition associated with the wavelet coefficients. Without loss of generality the nodes are located at the center of each cell. The solid lines show the connectivity relationship between the cells.

A pseudo code implementation of the adjacency search algorithm is given in Fig. 11. Note that the subroutine Reconstruct_Scan is called recursively to identify independent cells intersecting with the set $\mathcal{N}(\mathsf{x}_0, r_j)$. After a cell identified as an independent cell, the adjacency search algorithm continues on establishing the links from the current cell to adjacent independent cells along Left, Left-Top, Top, and Right-Top



(a) Search top-left (b) Search top

Figure 9: Refined adjacency search algorithm.

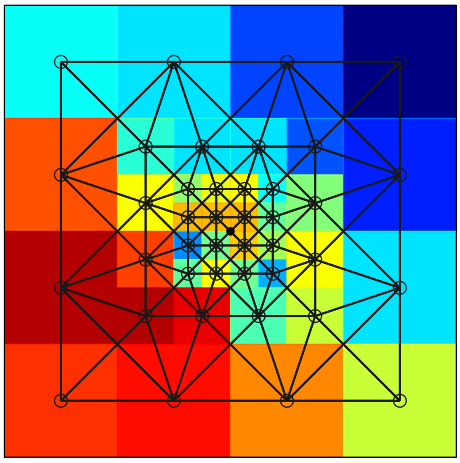


Figure 10: Connectivity relationship constructed from the multiresolution cell decomposition over three levels.

direction using the recursive refinements of search level as shown in Figs. 12 and 13.

Because the adjacency search algorithm uses recursive calls in both identification of cells and establishing links, obtaining the computational complexity for analytical expression is a non-trivial task. Rather, we estimate the computational complexity numerically. Suppose the input data to the algorithm are given by an $n \times n$ square image. The computational complexity of fast lifting wavelet transform is known to be $\mathcal{O}(N)$[145], where $N = n^2$. It should be noted that the computational cost can be tailored to the available computational resources of the agent due to multiresolution synthesis, however, it is rational to relate the computational complexity with the input data size N. For this purpose, we assumed two resolution levels, coarser level J_{rmmin} far away from the agent and finer level $J_{\max}$ close to the agent. Figure 14(a) shows the computational time in terms of various data size N in conjunction with a set of different sizes of high resolution window r. From Fig. 14 the numerical computational complexity of the adjacency search algorithm is deduced as a linear relationship with respect to the input data size N, or $\mathcal{O}(N)$. In contrast, Fig. 15 shows the computational complexity of the algorithm with respect to the window size r, which turns out to be a quadratic relationship $\mathcal{O}(r^2)$.

2.3.2 Cost assignment for $\mathcal{A}^*$ search

The $\mathcal{A}^*$ algorithm is a graph search algorithm that finds a path from an initial node to the goal node in the graph. The algorithm utilizes a *heuristic estimate* $h(v)$ that ranks each node v by a best cost estimate to reach the goal from the current node[49]. The algorithm visits the nodes in the order of the heuristic estimate, so the $\mathcal{A}^*$ algorithm is known as a best-first search algorithm. The key element of the $\mathcal{A}^*$ algorithm is that it expands each node from the priority queue that is ordered by (lower value has higher priority)

$$f(v) = g(v) + h(v), \tag{19}$$

```
BEGIN ADJACENCY SEARCH ALGORITHM
{
    Initialize graph structure 𝒢 = (V, E);
    𝒞_d ← Compute 2D FLWT(𝒲, J_min);
    (𝒦(J_min), ℒ(J_min)) ← 𝒩(x_0, r_{J_min});
    for (k = 0 : k_{J_min}) {
        for (ℓ = 0 : ℓ_{J_min}) {
            if (k, ℓ) ∈ (𝒦(J_min), ℒ(J_min))
                Reconstruct_Scan(k, ℓ, J_min, 𝒞_d);
            else
                AddNode_EstablishLinks(k, ℓ, J_min, 𝒞_d);
        }
    }
}
END ADJACENCY SEARCH ALGORITHM

Reconstruct_Scan(k, ℓ, j, 𝒞_d)
{
    𝒞_d ← Compute 2D Inverse FLWT(𝒞_d(k, ℓ; j));
    (𝒦(j + 1), ℒ(j + 1)) ← 𝒩(x_0, r_{j+1});
    for (k̄ = 2k : 2k + 1) {
        for (ℓ̄ = 2ℓ : 2ℓ + 1) {
            if (k̄, ℓ̄) ∈ (𝒦(j + 1), ℒ(j + 1))
                Reconstruct_Scan(k̄, ℓ̄, j + 1, 𝒞_d);
            else
                AddNode_EstablishLinks(k̄, ℓ̄, j + 1, 𝒞_d);
        }
    }
}

AddNode_EstablishLinks(k, ℓ, j, 𝒞_d)
{
    /* Add Node */
    v ← c^j_{k,ℓ} ∼ a_{j,k,ℓ};
    V(𝒢) ← v;
    Set d^i_{j',k,ℓ} ← 0 ,    ∀j' ≥ j;

    /* Establish Links */
    LinkLeft(k, ℓ, j, 𝒞_d);
    LinkLeftTop(k, ℓ, j, 𝒞_d);
    LinkTop(k, ℓ, j, 𝒞_d);
    LinkTopRight(k, ℓ, j, 𝒞_d);
}
```

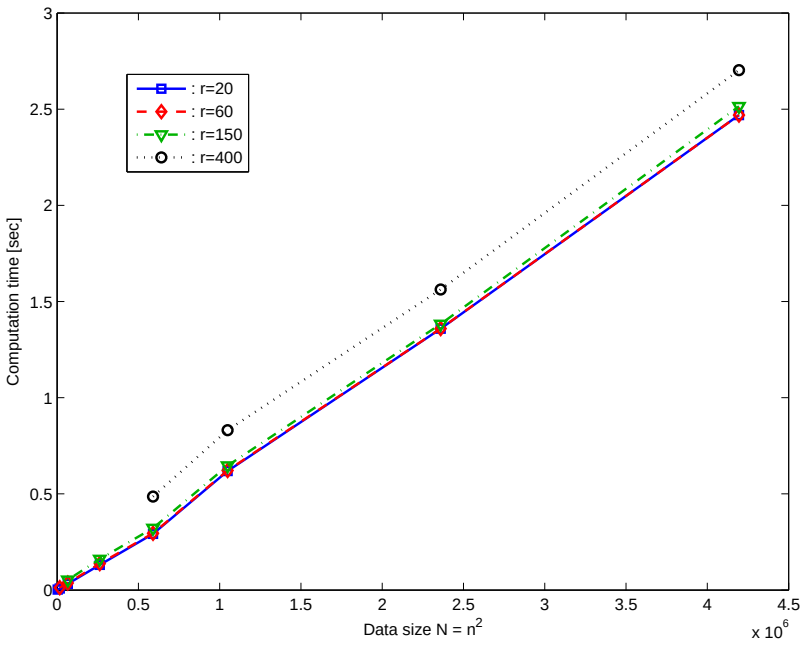
Figure 11: Pseudo-code implementation of the adjacency search algorithm.

```
LinkLeft(k, ℓ, j, C_d)
{
    c^j_{k,ℓ} ← Get current cell(k, ℓ, j, C_d);
    j_s ← Determine basic search level(c^j_{k,ℓ});
    (k_s, ℓ_s) ← Get parent cell index(c^j_{k,ℓ}, j_s);
    k_s = k_s − 1;    /* To examine left adjacent cell */
    LinkLeftRecurrence(c^j_{k,ℓ}, k_s, ℓ_s, j_s);
}
LinkLeftRecurrence(c^j_{k,ℓ}, k_s, ℓ_s, j_s)
{
    v ← c^j_{k,ℓ};
    if (d^1_{j_s,k_s,ℓ_s} = 0)    /* Horizontal detail */
        /* Adjacent independent cell */
        u ← c^{j_s}_{k_s,ℓ_s} ∼ a_{j_s,k_s,ℓ_s};
        E(G) ← (u, v), (v, u);  /* Bidirectional links */
    else
        /* Refine Search */
        j_s ← j_s + 1;
        (k_s, ℓ_s) ← Get refined left adjacent cell index(c^j_{k,ℓ}, j_s);
        LinkLeftRecurrence(c^j_{k,ℓ}, k_s, ℓ_s, j_s);
}

LinkLeftTop(k, ℓ, j, C_d)
{
    c^j_{k,ℓ} ← Get current cell(k, ℓ, j, C_d);
    j_s ← Determine basic search level(c^j_{k,ℓ});
    (k_s, ℓ_s) ← Get parent cell index(c^j_{k,ℓ}, j_s);
    k_s = k_s − 1; ℓ_s = ℓ_s − 1;    /* To examine left-top adjacent cell */
    LinkLeftTopRecurrence(c^j_{k,ℓ}, k_s, ℓ_s, j_s);
}
LinkLeftTopRecurrence(c^j_{k,ℓ}, k_s, ℓ_s, j_s)
{
    v ← c_{j,k,ℓ};
    if (d^3_{j_s,k_s,ℓ_s} = 0)    /* Diagonal detail */
        /* Adjacent independent cell */
        u ← c^{j_s}_{k_s,ℓ_s} ∼ a_{j_s,k_s,ℓ_s};
        E(G) ← (u, v), (v, u);  /* Bidirectional links */
    else
        /* Refine Search */
        j_s ← j_s + 1;
        (k_s, ℓ_s) ← Get refined left-top adjacent cell index(c^j_{k,ℓ}, j_s);
        LinkLeftTopRecurrence(c^j_{k,ℓ}, k_s, ℓ_s, j_s);
}
```

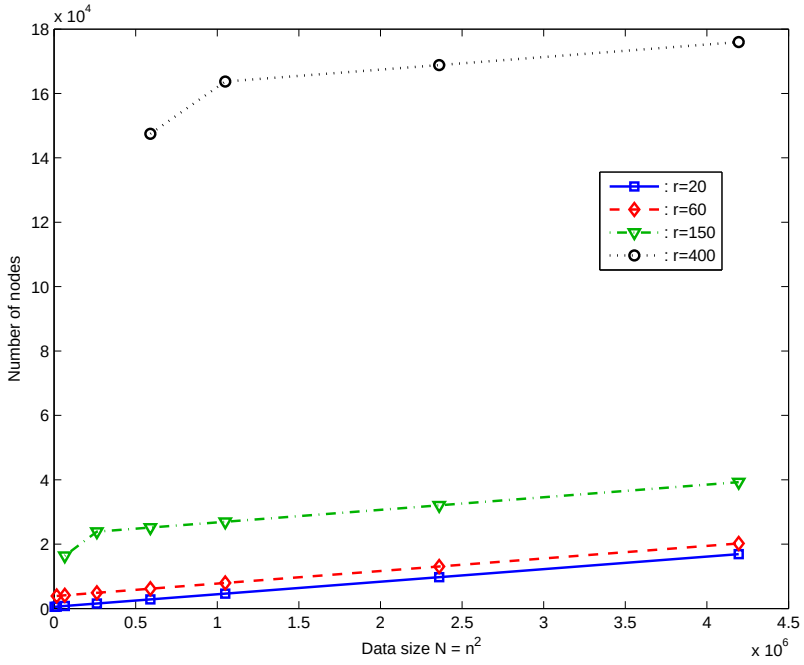
Figure 12: Pseudo-code implementation of the adjacency search algorithm: Recursive link connection.

```
LinkTop(k, ℓ, j, C_d)
{
    c^j_{k,ℓ} ← Get current cell(k, ℓ, j, C_d);
    j_s ← Determine basic search level(c^j_{k,ℓ});
    (k_s, ℓ_s) ← Get parent cell index(c^j_{k,ℓ}, j_s);
    ℓ_s = ℓ_s − 1;    /* To examine top adjacent cell */
    LinkTopRecurrence(c^j_{k,ℓ}, k_s, ℓ_s, j_s);
}
LinkTopRecurrence(c^j_{k,ℓ}, k_s, ℓ_s, j_s)
{
    v ← c^j_{k,ℓ};
    if (d^2_{j_s,k_s,ℓ_s} = 0)    /* Vertical detail */
        /* Adjacent independent cell */
        u ← c^{j_s}_{k_s,ℓ_s} ∼ a_{j_s,k_s,ℓ_s};
        E(G) ← (u, v), (v, u);  /* Bidirectional links */
    else
        /* Refine Search */
        j_s ← j_s + 1;
        (k_s, ℓ_s) ← Get refined top adjacent cell index(c^j_{k,ℓ}, j_s);
        LinkTopRecurrence(c^j_{k,ℓ}, k_s, ℓ_s, j_s);
}

LinkRightTop(k, ℓ, j, C_d)
{
    c^j_{k,ℓ} ← Get current cell(k, ℓ, j, C_d);
    j_s ← Determine basic search level(c^j_{k,ℓ});
    (k_s, ℓ_s) ← Get parent cell index(c^j_{k,ℓ}, j_s);
    k_s = k_s + 1; ℓ_s = ℓ_s − 1;    /* To examine right-top adjacent cell */
    LinkRightTopRecurrence(c^j_{k,ℓ}, k_s, ℓ_s, j_s);
}
LinkRightTopRecurrence(c^j_{k,ℓ}, k_s, ℓ_s, j_s)
{
    v ← c_{j,k,ℓ};
    if (d^3_{j_s,k_s,ℓ_s} = 0)    /* Diagonal detail */
        /* Adjacent independent cell */
        u ← c^{j_s}_{k_s,ℓ_s} ∼ a_{j_s,k_s,ℓ_s};
        E(G) ← (u, v), (v, u);  /* Bidirectional links */
    else
        /* Refine Search */
        j_s ← j_s + 1;
        (k_s, ℓ_s) ← Get refined right-top adjacent cell index(c^j_{k,ℓ}, j_s);
        LinkRightTopRecurrence(c^j_{k,ℓ}, k_s, ℓ_s, j_s);
}
```

Figure 13: Pseudo-code implementation of the adjacency search algorithm: Recursive link connection (continued).

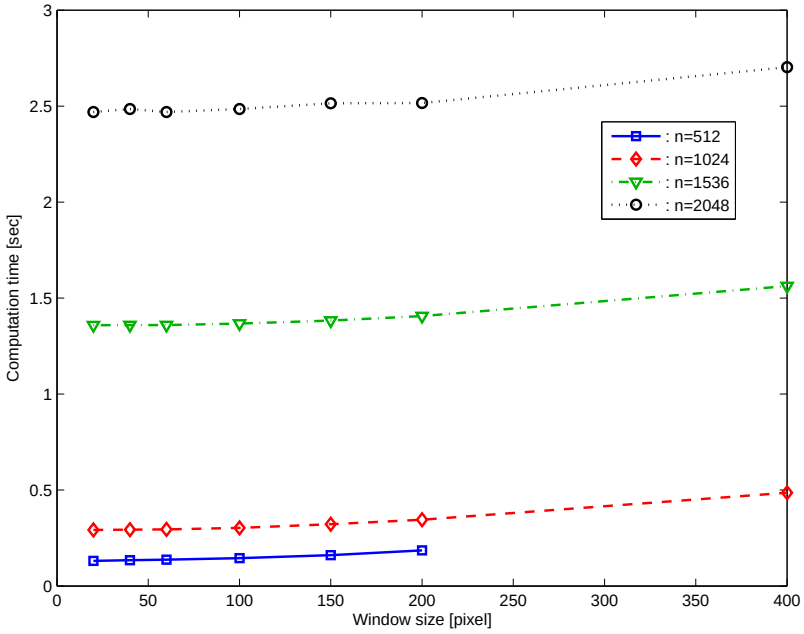


(a) Computational time

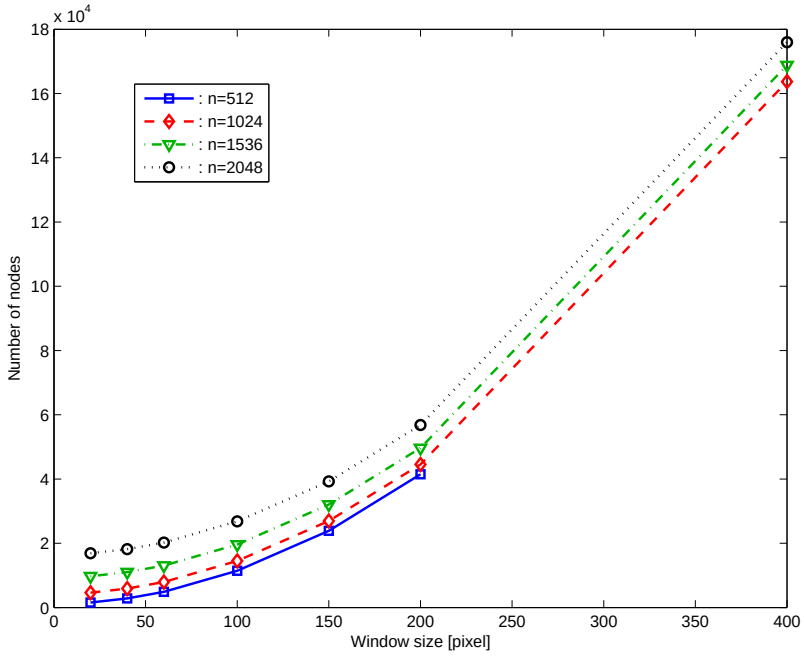


(b) Number of nodes

Figure 14: Computational cost for the adjacency search algorithm in terms of data size.



(a) Computational time



(b) Number of nodes

Figure 15: Computational cost for the adjacency search algorithm in terms of window size.

where the cost $g(v)$ is the actual cost of the path up to v, i.e. the sum of the edge costs from the initial node, and $h(v)$ is the heuristic estimate at v. When a node u is expanded, the adjacent nodes to the current node are exploited. Let v be the adjacent node, then it follows that we evaluate the actual cost $g(v)$ to see if the transition from u to v results in lower cost than any other transitions to v. The algorithm then sets a back-pointer $\pi(v)$ by its preceding node u. This process iterates until the goal node is reached and no other nodes have a lower cost to the goal.

The $\mathcal{A}^*$ algorithm is complete in the sense that it is always guaranteed to find a solution if a solution exists. In addition, if the heuristic function $h(v)$ is admissible, that is, it uses an underestimate of the actual cost of reaching the goal, then $\mathcal{A}^*$ is optimal. Details about the implementation of the $\mathcal{A}^*$ algorithm can be found, for instance, in Ref. [32].

To the cell decomposition (18) we associate each node $v \in \mathcal{G}$ o a cell $c^j_{k,\ell}$. Moreover, since $\mathcal{G}$ is a topological graph, we may associate each node v with some point $\mathsf{x} \in c^j_{k,\ell}$. Without loss of generality, we choose the center of the cell. Let $\mathsf{cell}_{\mathcal{G}}(v)$ denote the center of the corresponding cell. If $\mathsf{x} \in c^j_{k,\ell}$ we will write $v = \mathsf{node}_{\mathcal{G}}(\mathsf{x})$.

To each directed edge (u, v) of $\mathcal{G}$ we assign an edge cost, given as

$$\mathcal{J}(u, v) = \mathsf{rm}(\mathsf{cell}_{\mathcal{G}}(v)) + \alpha \|\mathsf{cell}_{\mathcal{G}}(u) - \mathsf{cell}_{\mathcal{G}}(v)\|_2, \tag{20}$$

where $\alpha \geq 0$ is a weight constant. The first term in (20) is proportional to the probability that the target node is close to obstacles, while the second term penalizes the (Euclidean) distance between $\mathsf{cell}_{\mathcal{G}}(u)$ and $\mathsf{cell}_{\mathcal{G}}(v)$.

Suppose now that we are given a path of $q + 1$ consecutive, adjacent nodes in $\mathcal{G}$ as follows

$$\mathcal{P} = (v_0, v_1, \cdots, v_q). \tag{21}$$

We can then assign a traversal cost to each node in the path $\mathcal{P}$, induced by

$$g(v_i) = g(v_{i-1}) + \mathcal{J}(v_{i-1}, v_i), \qquad i = 1, \cdots, q. \tag{22}$$

The value of $g(v_k)$ represents the (accumulated) cost of the path from v_0 to v_k ($k \leq q$), i.e. the weight of the edges followed up to v_k. We use the following heuristic estimate

$$h(v) = \|\mathsf{cell}_{\mathcal{G}}(v) - \mathsf{cell}_{\mathcal{G}}(v_f)\|_\infty, \tag{23}$$

where $v_f = \mathsf{node}_{\mathcal{G}}(\mathsf{x}_f)$.

The $\mathcal{A}^*$ algorithm then finds a path that minimizes the cost in (22) to the goal node, or determines that such a path does not exist.

2.4 Multiresolution Path Planning

2.4.1 Multiresolution path planning algorithm

The proposed multiresolution path planning algorithm proceeds as follows. Starting from $\mathsf{x}(t_0) = \mathsf{x}_0$ at time $t = t_0$, we construct using the approach of Section 2.2 a cell decomposition $\mathcal{C}_d(t_0)$ of $\mathcal{W}$. A topological graph, and the adjacency list of its nodes are obtained using the approach of Section 2.3. Let the corresponding graph be $\mathcal{G}(t_0)$ and let $v_1^0 \in \mathcal{G}(t_0)$ and $v_f^0 \in \mathcal{G}(t_0)$ be the initial and the goal nodes such that $v_1^0 = \mathsf{node}_{\mathcal{G}(t_0)}(\mathsf{x}_0)$ and $v_f^0 = \mathsf{node}_{\mathcal{G}(t_0)}(\mathsf{x}_f)$, respectively. Using the $\mathcal{A}^*$ algorithm we compute a path $\mathcal{P}(t_0)$ of free and mixed nodes from v_1^0 to v_f^0 in $\mathcal{G}(t_0)$ assuming that such a path exists. Let $\mathcal{P}(t_0)$ be given by an ordered sequence of $l_0 + 1$ nodes as follows

$$\mathcal{P}(t_0) = (v_0^0,\ v_1^0, \cdots,\ v_{l_0-1}^0,\ v_{l_0}^0 = v_f^0). \tag{24}$$

It is assumed that v_1^0 is a free node owing to the high resolution representation of $\mathcal{W}$ close to x_0. The agent subsequently moves from v_0^0 to v_1^0. Let now t_1 be the time the agent is at the location $\mathsf{x}(t_1) = \mathsf{cell}_{\mathcal{G}(t_0)}(v_1^0)$ and let $\mathcal{C}_d(t_1)$ be the multiresolution cell decomposition of $\mathcal{W}$ around $\mathsf{x}(t_1)$ with a corresponding topological graph $\mathcal{G}(t_1)$. Applying again the $\mathcal{A}^*$ algorithm we compute a (perhaps new) path in $\mathcal{G}(t_1)$ from $v_0^1 = \mathsf{node}_{\mathcal{G}(t_1)}(\mathsf{x}(t_1))$ to $v_f^1 = \mathsf{node}_{\mathcal{G}(t_1)}(\mathsf{x}_f)$ if such a path exists. Let $\mathcal{P}(t_1)$ be given by the ordered sequence of $l_1 + 1$ nodes as follows

$$\mathcal{P}(t_0) = (v_0^1,\ v_1^1, \cdots,\ v_{l_1-1}^1,\ v_{l_1}^1 = v_f^1). \tag{25}$$

BEGIN PATH PLANNING ALGORITHM
{
$i = 0$;
$\mathsf{x}(t_i) \leftarrow \mathsf{x}_0$;
`while` $\|\mathsf{x}(t_i) - \mathsf{x}_f\|_\infty \geq 1/2^{J_{\max}}$
{
`compute` $\mathsf{rm}(\mathsf{x}, i)$ for all $\mathsf{x} \in \mathcal{W}$;
`construct` $\mathcal{C}_d(i)$ at level $J_{\min}$;
`construct` $\mathcal{G}(i) = \big(E(i), V(i)\big)$;
$v_1^i \leftarrow \mathsf{node}_{\mathcal{G}(i)}\big(\mathsf{x}(t_i)\big)$;
$v_f^i \leftarrow \mathsf{node}_{\mathcal{G}(i)}(\mathsf{x}_f)$;
$\mathcal{P}(i) \leftarrow \mathtt{Astar}\big(v_1^i, v_f^i, \mathcal{G}(i)\big)$;
`if` $\mathcal{P}(i) = \varnothing$
`report FAILURE; break;`
$\mathsf{x}(t_{i+1}) \leftarrow \mathsf{cell}_{\mathcal{G}(i)}(v_2^i)$;
`Move to` $\mathsf{x}(t_{i+1})$;
$i \leftarrow i + 1$;
}
}
END PATH PLANNING ALGORITHM

Figure 16: Pseudo-code implementation of proposed multiresolution path planning scheme.

The agent subsequently moves to v_1^1 at location $\mathsf{x}(t_2) = \mathsf{cell}_{\mathcal{G}(t_1)}(v_1^1)$ at time t_2.

In general, assume the agent is at location $\mathsf{x}(t_i)$ at time t_i. We construct a multiresolution decomposition $\mathcal{C}_d(t_i)$ of $\mathcal{W}$ around $\mathsf{x}(t_i)$ with a corresponding graph $\mathcal{G}(t_i)$. The $\mathcal{A}^*$ algorithm yields a path $\mathcal{P}(t_i)$ in $\mathcal{G}(t_i)$ of length $l_i + 1$,

$$\mathcal{P}(t_i) = (v_0^i,\ v_1^i, \cdots,\ v_{l_i-1}^i,\ v_{l_i}^i = v_f^i), \tag{26}$$

where $v_0^i = \mathsf{node}_{\mathcal{G}(t_i)}(\mathsf{x}(t_i))$ and $v_f^i = \mathsf{node}_{\mathcal{G}(t_i)}(\mathsf{x}_f)$. This iteration process terminates at some time t_f when $\|\mathsf{x}(t_f) - \mathsf{x}_f\|_\infty < 1/2^{J_{\max}}$. At the last step the agent moves from $\mathsf{x}(t_f)$ to x_f. A pseudo code implementation of the multiresolution path planning algorithm is given in Fig. 16. Note that the actual path followed by the agent is given by the sequence of nodes $\big\{\mathsf{node}_{\mathcal{G}(t_0)}\big(\mathsf{x}(t_0)\big), \mathsf{node}_{\mathcal{G}(t_1)}\big(\mathsf{x}(t_1)\big), \cdots, \mathsf{node}_{\mathcal{G}(t_f)}\big(\mathsf{x}(t_f)\big)\big\}$.

2.4.2 $\mathcal{D}^*$-lite path planning algorithm

The $\mathcal{D}^*$ algorithm has been originally proposed by Stentz[135, 136] for planning a path in unknown or partially known environment. Prior to $\mathcal{D}^*$, several replanning strategies have been proposed to solve dynamic planning problems for locally-directed wandering[110], local modification of initial path[72], and obstacle perimeter detouring[86]. Although these methods are complete, they are suboptimal and computationally inefficient. On the contrary, $\mathcal{D}^*$ produces an optimal path by adopting an efficient incremental search scheme to reduce the time required to replan. In particular, $\mathcal{D}^*$ is more appropriate when dealing with an environment having a large number of states, by reusing information from the previous search to find the solution at the next step. Koenig and Likhachev introduced Lifelong Planning $\mathcal{A}^*$ (LPA)[70] which employs heuristic estimate like $\mathcal{A}^*$, while reusing information from previous searches to find a solution much faster than solving each iteration from scratch. Furthermore, Koenig and Likhachev presented the $\mathcal{D}^*$-lite algorithm, derived from the LPA algorithm, which implements the same planning strategy as $\mathcal{D}^*$ but is algorithmically different. The $\mathcal{D}^*$-lite algorithm simplifies the maintenance of priority queues, thus it does not use complicated conditional statements, thus ending up with shorter codes than the original $\mathcal{D}^*$ algorithm. In the sequel, we employ the $\mathcal{D}^*$-lite algorithm to the path planning problem on a non-trivial environment. By comparing the $\mathcal{D}^*$-lite algorithm with the multiresolution path planning algorithm, we discuss the benefits and shortfalls of using the multiresolution path planning algorithm over the $\mathcal{D}^*$-lite algorithm.

We apply the Haar wavelet transform up to resolution level $J \geq J_{\min}$ to obtain the wavelet decomposition of rm, given by

$$\mathsf{rm}(x,y) = \sum_{k,\ell=0}^{2^J-1} a_{J,k,\ell}\Phi_{J,k,\ell}(x,y) + \sum_{i=1}^{3}\sum_{j=J}^{N-1}\sum_{k,\ell=0}^{2^j-1} d^i_{j,k,\ell}\Psi^i_{j,k,\ell}(x,y). \tag{27}$$

A uniform cell decomposition $\mathcal{C}_d^J$ at level J on $\mathcal{W}$ is induced from Eq. (27), and is

comprised of cells $c^J_{k,\ell}$ of dimension $1/2^J \times 1/2^J$. We adopt the eight-connectivity relationship between the cells. The connectivity relationship is easily found by bookkeeping the location of each cell through the indices k and ℓ. It should be noted that the adjacency relationship will remain the same throughout the replanning, but the edge costs will change incrementally to incorporate the information from the previous step.

Suppose the agent is equipped only with a proximity sensor that senses the environment close to the current location with high accuracy. That is, the sensor provides information about classification of the neighboring environment by a free region or obstacle region. Let $\mathcal{S}(i)$ be the known region up to $t = t_i$ using a sensor with the range r_J, defined by

$$\mathcal{S}(i) = \mathcal{S}(i-1) \cup \mathcal{N}(\mathsf{x}_i, r_J) \tag{28}$$

where x_i is the current location of the agent at $t = t_i$ and $\mathcal{N}(\mathsf{x}_i, r_J)$ represents the effective sensory region at that moment. In Eq. (28) it is assumed that the agent navigates an initially unknown environment while updating the map from the collected information. In order to take this process into consideration for replanning, we assign a conditional cost to each edge (u, v) which depends on the relative location of the edges to $\mathcal{S}$ as follows,

$$\mathcal{J}(u,v) = \begin{cases} \mathsf{rm}(\mathsf{cell}_{\mathcal{G}}(v)) + \|\mathsf{cell}_{\mathcal{G}}(u) - \mathsf{cell}_{\mathcal{G}}(v)\|_2, & \text{if } u,\ v \in \mathcal{S}, \\ \|\mathsf{cell}_{\mathcal{G}}(u) - \mathsf{cell}_{\mathcal{G}}(v)\|_2, & \text{if } u,\ v \in \mathcal{W} \setminus \mathcal{S}. \end{cases} \tag{29}$$

It follows that for the edges outside $\mathcal{S}$ we simply impose the traversal cost between nodes considering the uniform size of cells. With the traversal cost, a general path planning algorithm such as Dijkstra's or $\mathcal{A}^*$ simply computes a shortest path from an initial node to the goal node which might pass through obstacles outside $\mathcal{S}$. Nevertheless, whenever the map is updated using contingent information from the sensor, we accordingly update the corresponding edge costs by appending the obstacle cost to each edge as given in (29).

The main $\mathcal{D}^*$-lite path planning algorithm proceeds as follows. From the uniform cell decomposition and the corresponding graph, we solve for an initial path from $v_0 = v_{\text{start}}$ to v_{goal} assuming only the distance cost for edge weights. Let v_1 be the node next to v_0 in the path. The agent subsequently moves from v_0 to v_1. At time $t = t_1$ when the agent is located at v_1, the algorithm continues to scan the graph for changed edge costs. If any edge costs have changed, then the algorithm updates the corresponding edge weights. Subsequently, a new path is computed from v_1 to v_{goal}, by incorporating the updated edge weights. It should be noted that if no edge costs have changed the agent moves to the successive node v' in the previous path that has the minimum cost $\mathcal{J}(v_{\text{last}}, v') + g(v')$.

Similar to $\mathcal{A}^*$, the $\mathcal{D}^*$-lite algorithm also incorporates a heuristic estimate to choose the nodes from a priority queue. However, as the agent detects changes in the edge costs, the priority queue should be reordered to render itself consistent. This might be an expensive task, so instead of reordering the priority queue every time, Koenig and Likhachev utilizes a *dynamic heuristic constant* k_m[71] to keep the priority queue unaltered regardless of the change of the edge costs. This reduces the computational overhead, resulting in faster execution. The iteration terminates at some time t_l when the goal node is reached. A pseudo code implementation of the $\mathcal{D}^*$-lite incremental path planning algorithm is given in Fig. 16. Note that the actual path followed by the agent is given by the sequence of nodes $\{v_0 = v_{\text{start}}, v_1, \cdots, v_{l-1}, v_l = v_{\text{goal}}\}$.

2.5 Simulation Results

2.5.1 Simulation results for the proposed algorithm

In this section we present simulation results of the proposed algorithm for a non-trivial scenario. The environment $\mathcal{W}$ is an actual topographic (elevation) map of a US state, shown in Fig. 18. The environment is assumed to be square of dimension

```
BEGIN D*-LITE PATH REPLANNING ALGORITHM
{
    i = 0;
    compute rm(x) for all x ∈ W;
    construct C_d at level J_max;
    construct G = (E, V);
    v_start ← node_G(x_0);
    v_goal ← node_G(x_f)                                                    ;
    Initialize();
    ComputeShortestPath(v_start, v_goal, G);
    v_i ← v_start;
    v_last ← v_i;
    while (v_i ≠ v_goal)
    {
        i ← i + 1;
        v_i = argmin_{v' ∈ Adj(v_last)} ( J(v_last, v') + g(v') );
        Move to v_i;
        Scan graph for changed edge costs;
        If any edge costs changed;
        {
            k_m ← k_m + h(v_last, v_i);
            v_last ← v_i;
            For all directed edges (u, v) ∈ E with changed edge costs
            {
                Update the edge cost J(u, v);
                UpdateVertex(v);
            }
            ComputeShortestPath(v_i, v_goal, G);
        }
    }
}
END D*-LITE PATH REPLANNING ALGORITHM
```

Figure 17: Pseudo-code implementation of $\mathcal{D}^*$-lite path planning scheme.

128×128 units. Hence the finest possible resolution is $N = 7$. Taking into account the available memory of the micro-controller, we choose the fine level as $J_{\max} = 6$ and the coarse level as $J_{\min} = 3$. This makes the total number of nodes in the graph not to exceed the maximum count of 256 that corresponds to the maximum allowable variable size of the micro-controller. The ranges at distinct levels of resolution are selected $(r_6, r_5, r_4) = (8, 15, 30)$ units from the current location. It follows that the fine resolution at level $J = 6$ is used inside an area of 16×16 units around the current location of the agent, and the coarser resolutions at levels $J = 5$ and $J = 4$ are used inside 30×30 and 60×60 units.

The objective of the UAV is to follow a path from the initial position to the final position while circumventing the obstacles over a certain elevation threshold. Since the on-line path planning problem at the finest resolution is computationally prohibitive, the proposed algorithm accommodates the need of the on-line implementation for the micro-controller by limiting the amount of the information to process, thus computing an immediate path with high accuracy within the allowable time scale of the micro-controller.

The results from the multiresolution path planning algorithm are shown in Fig. 18. Specifically, Fig. 18 shows the evolution of the path at different time steps as the agent moves to the final destination. Figure 18(a) shows the agent's position at time step $t = t_5$ along with the best proposed path to the final destination by a dashed-dot line at that time. The actual path followed by the agent is drawn by a solid line. Similarly, Fig. 18(b) shows the agent's position at time step $t = t_{21}$. As seen in Fig. 18(c), the actual path followed by the agent differs from the one predicted in either Figs. 18(a) or 18(b). This is due to the fact that at time t_5 and t_{21} the agent does not have complete information for upcoming location up to confident level. In particular, Fig. 18(b) shows that as the agent gets closer to the obstacle, it recognizes the presence of obstacles and redirects the path to avoid the obstacle. Finally, the

agent reaches the final destination x_f in a collision free manner, as seen in Fig. 18(c).

2.5.2 Simulation results for the $\mathcal{D}^*$-lite algorithm

In this section we present simulation results of the $\mathcal{D}^*$-lite path planning algorithm for the same environment used in the previous simulation. It is assumed that the agent navigates over the unknown environment, while updating the map with the information gathered from a proximity sensor. We adopt a uniform cell decomposition of cell size the $J_{\max} = 6$ which is the same to the finest level used in the previous section. The range of the proximity sensor is chosen $r_6 = 7$, thus resulting in a high resolution window by a 7×7 square grid.

The results from the $\mathcal{D}^*$-lite path planning algorithm are shown in Fig. 19. Also, Fig. 19 shows the evolution of the path at different time steps as the agent moves to the final destination. At each step, the best proposed path is drawn by a dashed-dot line and the actual path followed by the agent is drawn by a solid line. As seen in Fig. 19(c), the actual path differs significantly from the one predicted in either Figs. 19(a) or 19(b). This is attributed to the fact that the environment is unknown a priori, and the initial path is computed using the distance cost outside the high resolution horizon. Hence, as shown in Fig. 19(a), the agent is unable to anticipate the existence of the obstacles outside the high resolution horizon. Nonetheless, as the agent gets closer to the obstacles, the $\mathcal{D}^*$-lite algorithm effectively replans the entire path avoiding the obstacles, reaching the final destination.

2.6 Comparison

The proposed multiresolution path planning algorithm was written in C code and implemented on an on-board autopilot equipped with a Rabbit RCM-3400 micro-controller. Because the micro-controller has limited computational resources (10,000 instructions per second, and 512 KB RAM for handling variables), the code has been written giving special attention not only to the accuracy of the output, but

Table 1: Computational cost of the proposed algorithm by the on-board autopilot.

Multiresolution cell decomposition using FLWT	452 [msec]
Construct the connectivity relationship and $\mathcal{G}$	292 [msec]
Compute a path using $\mathcal{A}^*$ employing the binary heap	202 [msec]
Average number of nodes of each $\mathcal{G}$	$\sim$200

also to the computational speed during implementation. Specifically, most of the computations for the proposed algorithm is done using integer arithmetic. Given a risk measure rm of integer samples, the integer fast lifting wavelet transform provides the approximation and detail coefficients that are used to construct the adjacency relationship between cells. The $\mathcal{A}^*$ algorithm is then called to find the shortest path in the graph associated with the wavelet decomposition.

Table 1 shows the computational cost of the proposed path planning algorithm using the on-board autopilot. One step of the path planning iteration takes 946 [msec] for execution. With the knowledge of the execution time of the proposed algorithm, we actually choose to implement the proposed path planning algorithm on-line every three seconds. Hence, the autopilot manages not only to execute the basic tasks such as data acquisition and processing, inner loops control, and etc., but also to plan a path in a seamless manner.

We compared the computational costs between the proposed multiresolution path planning algorithm and the $\mathcal{D}^*$-lite algorithm, using different simulation results for several cases. The simulations were carried out on an IBM-PC (Pentium M 2.0 GHz, 1 GB RAM), based on codes written in C for implementing both path planning algorithms. The proposed path planning algorithm accomplishes the path planning objective of reaching the goal in a fewer number of iterations, as shown in Table 2, than the $\mathcal{D}^*$-lite algorithm. This is due to the fact that the proposed algorithm effectively manages the information of the coarser resolutions so that a preferred path is computed over the approximation of $\mathcal{W}$. The $\mathcal{D}^*$-lite algorithm, however,

Table 2: The computational cost comparison between the multiresolution path planning v/s the $\mathcal{D}^*$-lite.

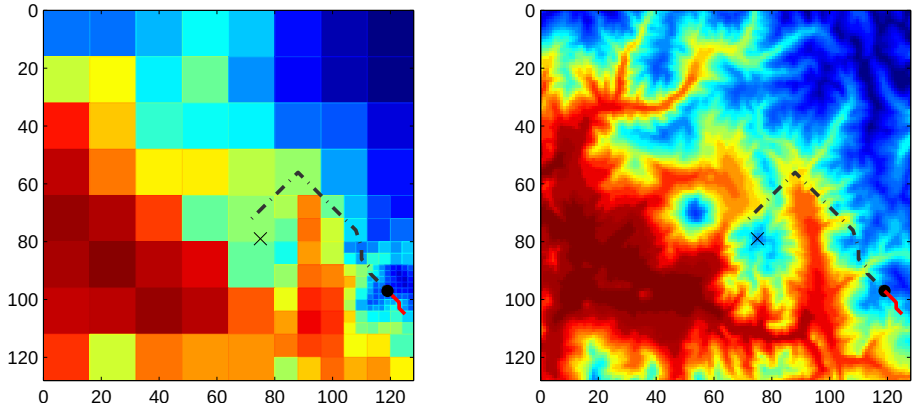
Items	Scenario I		Scenario II		Scenario III		Scenario IV		Scenario V	
	$\mathcal{D}^*$-lite	Wavelet	$\mathcal{D}^*$-lite	Wavelet	$\mathcal{D}^*$-lite	Wavelet	$\mathcal{D}^*$-lite	Wavelet	$\mathcal{D}^*$-lite	Wavelet
# iteration	35	31	93	50	61	40	123	52	44	43
# nodes in $\mathcal{G}$	4096	201	4096	194	4096	192	4096	185	4096	194
Data processing [msec]	2.03	2.03	2.03	2.03	2.03	2.03	2.03	2.03	2.03	2.03
Adjacency search [msec]	-	0.977	-	0.987	-	0.969	-	0.94	-	0.958
$\mathcal{A}^*$ search [msec]	-	0.1	-	0.125	-	0.094	-	0.138	-	0.066
Init. $\mathcal{D}^*$-lite search [msec]	1.87	-	2.03	-	2.03	-	1.87	-	2.03	-
$\mathcal{D}^*$-lite update [msec]	4.1	-	23.8	-	11.3	-	43.9	-	12.7	-
Total Comp. time [msec]	5.97	33.387	25.83	55.6	13.33	42.53	45.77	56.056	14.73	44.032
Computational cost (%)	17.8	100	46.46	100	31.35	100	81.65	100	33.45	100
Memory cost (%)	2037.8	100	2111.3	100	2133.3	100	2214.1	100	2111.3	100

relies on the information at finer resolution which is unveiled up to the current time, thus requiring the agent to explore the environment and to replan the path along the movement of the agent. In the worst case, the total number of iterations by the $\mathcal{D}^*$-lite algorithm increases significantly (e.g. Scenario IV) because of the existence of unknown obstacles.

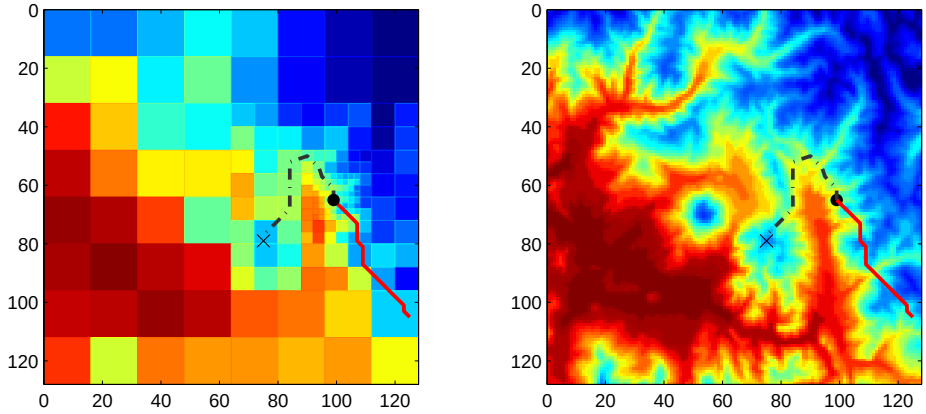
The total computation time of the proposed algorithm is obtained by adding the computation times throughout each iteration. For the $\mathcal{D}^*$-lite algorithm, the total computation time consists of the time for initializing and updating, which is shown to be smaller than that of the proposed algorithm. The $\mathcal{D}^*$-lite algorithm is computationally efficient in the sense that it reuses information from the previous step. In contrast, most of the computations in the proposed algorithm are devoted to the construction of the adjacency list at each planning step, as shown in Table 2. The performance of the proposed algorithm can thus be improved by using, say, four-connectivity instead of eight-connectivity during the adjacency search algorithm. This will possibly halve the computation time with little performance degradation. By the inherent benefit from the multiresolution decomposition, the proposed algorithm requires little memory as shown in Table 2 compared to $\mathcal{D}^*$-lite. For on-line, on-board path planning, the proposed algorithm has the advantages of scalability according to the available on-board computational resources.

2.7 Summary

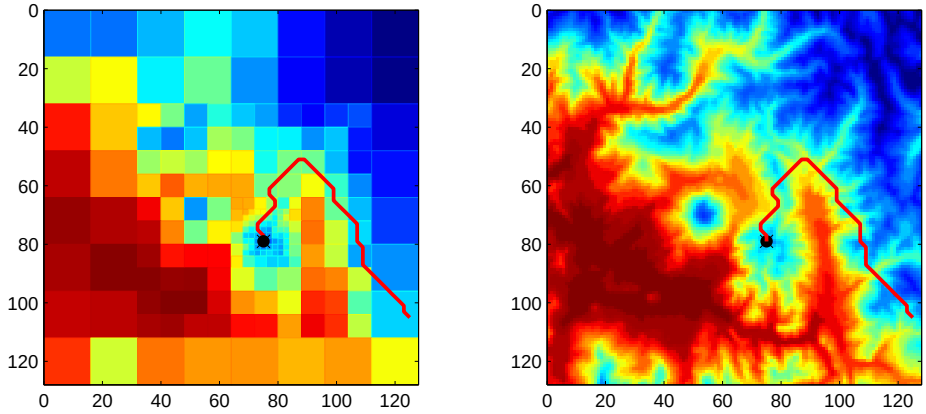
Autonomous path planning for small UAVs imposes severe restrictions on control algorithm development, stemming from the limitations imposed by the on-board hardware and the requirement for on-line implementation. In this chapter we have proposed a method to overcome this problem by using a new multiresolution path planning scheme. The algorithm computes at each step a multiresolution representation of the environment using the fast lifting wavelet transform. By utilizing most of integer arithmetic of FLWT, the computational cost is significantly reduced. The idea is to employ high resolution close to the agent (where is needed most), and a coarse resolution at large distances from the current location of the agent. As an added benefit, the connectivity relationship of the resulting cell decomposition can be computed directly from the nonzero detail coefficients of the wavelet transform. The algorithm is scalable and can be tailored to the available computational resources of the agent.



(a) $t = t_5$

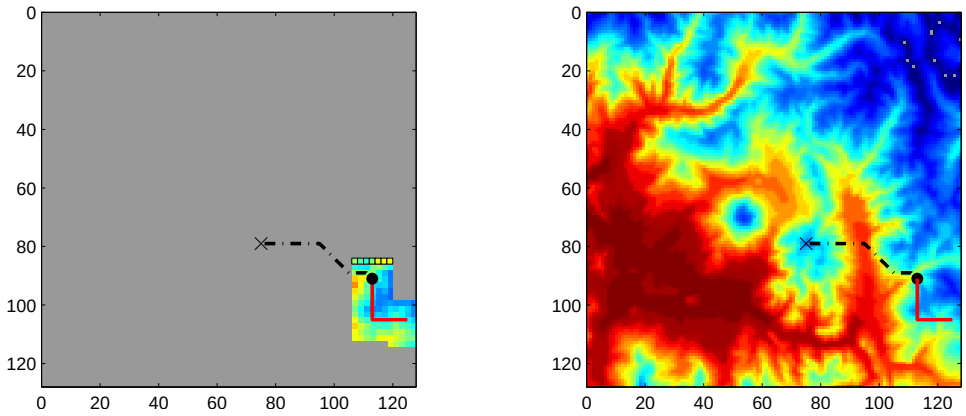


(b) $t = t_{21}$

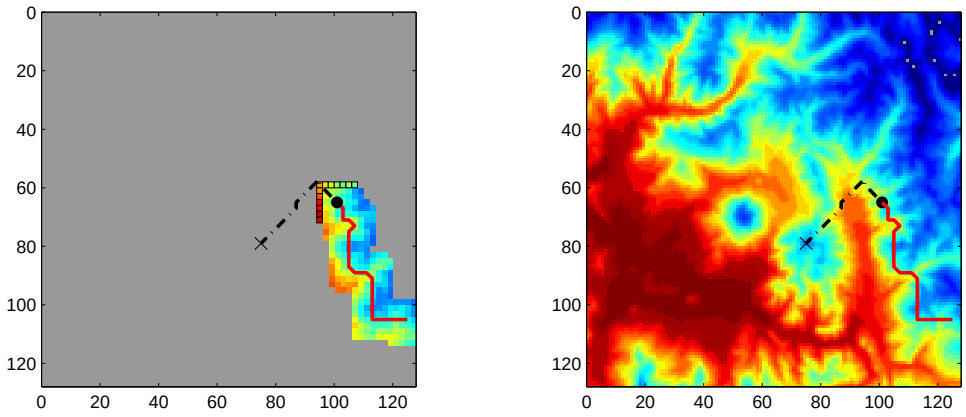


(c) $t = t_f$

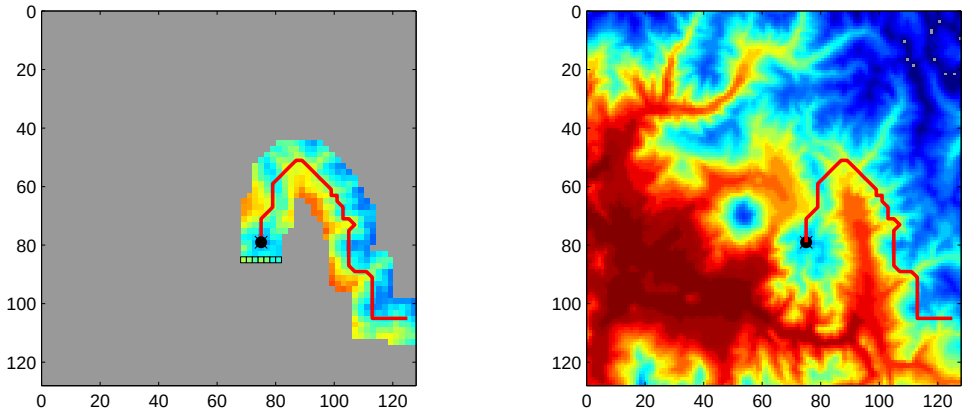
Figure 18: Path evolution and replanning. Dashed-dot lines represent the currently tentative optimal path obtained from the $\mathcal{A}^*$ algorithm, based on the available multiresolution approximation of the environment at different time steps. Solid lines reveal the actual path followed by the agent.



(a) $t = t_{14}$



(b) $t = t_{30}$



(c) $t = t_f$

Figure 19: Path evolution and replanning using the $\mathcal{D}^*$-lite algorithm. Dashed-dot lines show the currently tentative optimal path obtained from the $\mathcal{D}^*$-lite algorithm, based on the distance cost outside the high resolution area. The actual path followed by the agent is drawn by solid lines.

CHAPTER III

ON-LINE PATH SMOOTHING USING PATH TEMPLATES

3.1 Introduction

Guidance and navigation control of mobile agents has been an important research topic for several decades. In particular, for unmanned aerial vehicles (UAVs), the ability of fully automated guidance and navigation control allows the UAVs to accomplish missions under various circumstances with minimal human intervention. As a matter of fact, because of the stringent operational requirements and the restrictions imposed on UAVs by autonomy, safety, efficiency, etc., a complete solution to fully automated guidance and navigation control of UAVs is challenging. Many researchers suggest that breaking the problem into several subproblems such as path planning, trajectory smoothing, trajectory tracking, and etc., makes it easy to solve by a hierarchical control structure [93, 10, 94].

It is assumed that a planned path is given by a series of way-points from the top-level path planner. Traditionally, a simple implementation of the way-points tracking control is used for guidance purpose. A better implementation of the guidance and navigation control incorporates a smooth path by taking into account the dynamic constraints of UAVs. The way-points can then be connected to generate smooth path segments, which preserves the continuity of curvature between line and arc segments while minimizing the maximum curvature on the curve [61, 122]. In Ref. [6], the authors proposed a dynamic trajectory smoothing algorithm by which the path segments in straight-lines are smoothed to yield an extremal trajectory with explicit consideration of the kinematic constraint of a fixed-wing UAV.

In contrast to the explicit consideration of the kinematic constraint of the UAV, a spline-based path generation method has been widely adopted when computing the smooth, dynamically feasible trajectory for UAVs. A series of cubic splines was employed in Ref. [56] to connect straight line segments in a near-optimal manner. The authors in Ref. [147] presented an implicit time-parameterization of the trajectory using a B-spline representation. Designing an obstacle-avoiding B-spline path has been dealt with by Berglund et al.[15], whereas the real-time modifications of a spline path is proposed in Ref. [39]. The advantage of employing the (B-)splines in generating a smooth path is dictated by the fact that the path can be represented by using smaller number of parameters than using complete description of path. Accordingly, both for path optimization and for on-line implementation, it is straightforward to deal with small number of parameters when generating paths that are subject to the given obstacle constraints at the minimum computational cost.

The obstacle avoidance path planning problem using B-splines involves a constrained optimization such that the path should not only avoid forbidden regions but also become a flyable trajectory. In Ref. [87], a polygonal channel comprised of piecewise polylines serves as constraint equations while a B-spline curve is optimized using quadratic programming. This has been made possible by adopting tight linear envelopes for splines [88], by which a B-spline is represented as an approximate bounding polygon. In their approach, one dimensional B-splines are utilized to describe a smooth path subject to the channel constraints. In this paper, we extend the results in Ref. [88] in two dimensions, thus incorporating a two dimensional B-spline curve instead of a B-spline function. To this end, we formulate an optimization problem similar to the channel problem in Ref. [87] with constraints being given as geometric constraints.

Incorporating a high-level path planning algorithm such as $\mathcal{D}^*$-lite, we present a path smoothing algorithm using a set of path templates. Instead of smoothing the

entire path from an initial position to the goal position, we smooth the path segments over a finite planning horizon with respect to the current position of the UAV. This approach is somewhat similar to implementing a receding horizon concept in the path generation stage. To the problem of trajectory generation and collision avoidance, the receding horizon concept has been successfully implemented showing that it is an effective way to reduce the computational cost [43, 67, 129]. Each segment of B-spline curves are then stitched together to the B-spline curve that corresponds to the next path sequence, thus overall path remains smooth. Although the explicit dynamics of the UAV are not directly dealt with for smoothing the path segments, this approach has the advantage of having minimal on-line computational cost since most of computation is done off-line.

3.2 Tight Envelope for B-spline curves

3.2.1 Tight envelope for B-spline function

An one-dimensional spline function b is expressed by

$$b(u) = \sum_{j=0}^{m} b_j N_j^d(u) \tag{30}$$

where b_j are control points and $N_j^d(u)$ are the B-spline basis functions of degree d which are defined over a non-decreasing knot sequence $\{u_k\}$ such that $u_0 \leq u_1 \leq \cdots \leq u_{m+d+1}$. The number of knots is determined by the sum of the number of control points $(m+1)$ and the B-spline order $(d+1)$. The first and the last knots of the sequence should have multiplicity $(d+1)$ for a B-spline to pass the the first and the last control points, in such that $u_0 = u_1 = \cdots = u_d$ and $u_{m+1} = u_{m+2} = \cdots = u_{m+d+1}$, respectively. The B-spline basis functions are computed by the well-known *Cox-de*

Boor recursion formulas [34] from the degree 0 to d as follows,

$$N_j^0(u) = \begin{cases} 1 & \text{if } u_j \le u < u_{j+1}, \\ 0 & \text{otherwise,} \end{cases} \tag{31a}$$

$$N_j^d(u) = \frac{u - u_j}{u_{j+d} - u_j} N_j^{d-1}(u) + \frac{u_{u+d+1} - u}{u_{j+d+1} - u_{j+1}} N_{j+1}^{d-1}(u). \tag{31b}$$

The B-spline basis function has several useful properties. Among them, it is well known that B-spline basis function has local support [109], that is $N_j^d(u)$ is a non-zero polynomial over a knot span $[u_j,\ u_{j+d+1})$, or given any span $[u_j,\ u_{j+1})$, at most $(d+1)$ B-spline basis functions of degree d are non-zero. This local support property is important to curve design, since it allows to modify the B-spline curve locally without changing the entire shape. Figure 20 shows the cubic B-spline basis functions of degree $d = 3$ over the knot sequence $u \in [0,\ 1]$.

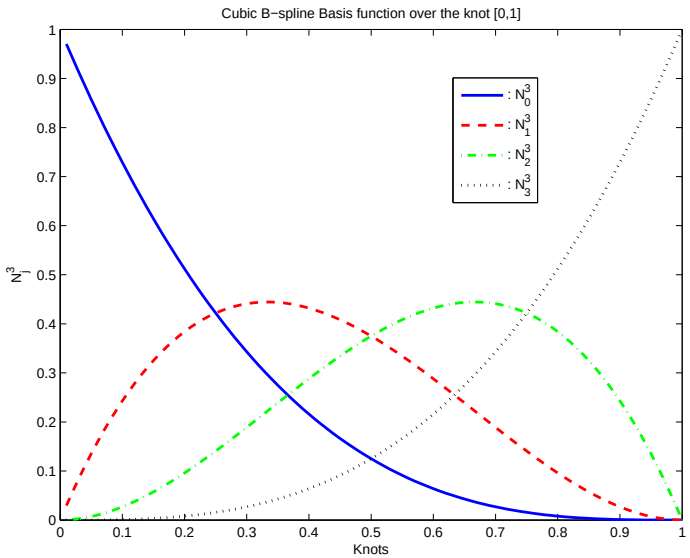


Figure 20: B-spline basis functions N_j^3 over the knot $u \in [0,\ 1]$.

Let the control polygon of the B-spline ℓ be defined by piecewise line segments connecting the control points b_j at the Greville abscissae[48],

$$u_j^* = \sum_{i=j+1}^{j+d} u_i/d, \tag{32}$$

such that at each Greville abscissae it satisfies $\ell(u_j^*) = b_j$. Accordingly, an envelope of the B-spline specifies a bound on the distance between b and its control polygon ℓ. This envelope should provide a good estimate of the shape of the B-spline by approximating the spline using less information. Although a number of bounds for splines are proposed in the literature, the bound derived in Ref. [99, 88] is known to be a tight, quantitative bound. Hence, it is possible to approximate the B-splines by piecewise linear envelopes, as the envelopes carry most of salient information about the curve itself. In light of this, the envelopes have advantage of being incorporated in the B-spline optimization problem, thus simplifying the analysis.

The envelopes in Ref. [88] are expressed in terms of the weighted second difference of the control points,

$$\Delta_2 b_j \triangleq b'_{j+1} - b'_j \qquad \text{where} \quad b'_j = \frac{b_j - b_{j-1}}{u_j^* - u_{j-1}^*}, \tag{33}$$

and by the non-negative and convex functions over the interval $[u_k^*, u_{k+1}^*]$ $(k = 0, 1, \cdots, m)$ as follows,

$$\beta_{ki} = \begin{cases} \sum_{j=i}^{\overline{k}} (u_j^* - u_i^*) N_j^d(u) & i > k, \\ \sum_{j=\underline{k}}^{i} (u_i^* - u_j^*) N_j^d(u) & j \leq k, \end{cases} \tag{34}$$

where, $\underline{k}$ and $\overline{k}$ denote the index of the first and the last B-spline basis functions which are nonzero over the corresponding interval, respectively. Subsequently, the distance between the spline function b and its control polygon ℓ is calculated as follows,

$$b - \ell = \sum_{i=\underline{k}}^{\overline{k}} \Delta_2 b_i \beta_{ki}. \tag{35}$$

Furthermore, by choosing the maximum and minimum variation of the weighted second difference as $\Delta_i^- = \min\{0, \Delta_2 b_i\}$, $\Delta_i^+ = \max\{0, \Delta_2 b_i\}$, we obtain the maximum offsets in both positive and negative direction with respect to the control polygon, which, in turn, become the upper and lower bounds of the spline function with respect

to the control polygon,

$$\ell + \sum_{i=\underline{k}}^{\overline{k}} \Delta_i^- \beta_{ki} \leq b \leq \ell + \sum_{i=\underline{k}}^{\overline{k}} \Delta_i^+ \beta_{ki}. \tag{36}$$

Since the β_{ki}'s are non-negative and convex function over the corresponding interval $[u_k^*,\ u_{k+1}^*]$, the maximum function values occur at each end of the interval, i.e. at each Greville abscissae. (See Fig. 21). Then the piecewise linear functions $\underline{e}$ and

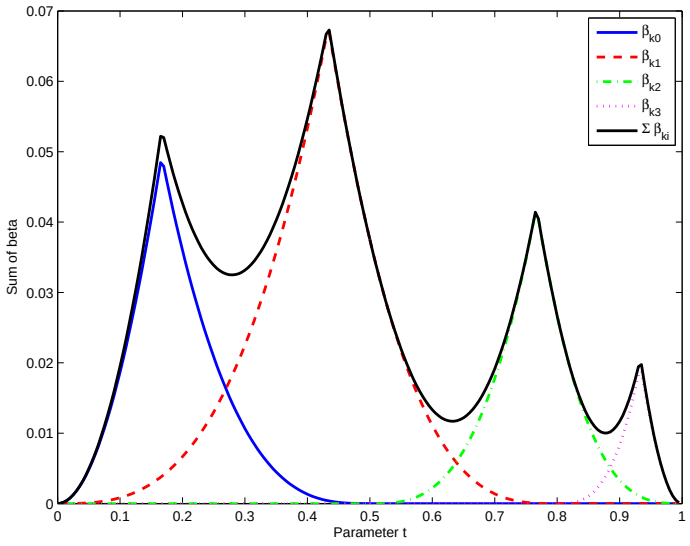


Figure 21: Non-negative and convex functions β_{ki}.

$\overline{e}$ that interpolate their values at each Greville abscissa can be employed to provide tight envelopes of the spline function,

$$\begin{aligned} \underline{e} &= \ell + \mathcal{L}\left(\sum \Delta_i^- \beta_{ki}(u_k^*), \sum \Delta_i^- \beta_{k+1,i}(u_{k+1}^*) \right), \\ \overline{e} &= \ell + \mathcal{L}\left(\sum \Delta_i^+ \beta_{ki}(u_k^*), \sum \Delta_i^+ \beta_{k+1,i}(u_{k+1}^*) \right), \end{aligned} \tag{37}$$

where $\mathcal{L}(\cdot, \cdot)$ denotes a linear interpolation between two values. Therefore, the maximal bounds from the B-spline function to its control polygon are obtained in a simple form,

$$\underline{e} \leq b \leq \overline{e}. \tag{38}$$

Figure 22 shows a cubic B-spline function b over the knot sequence $u \in [0,\ 1]$. The bounding envelopes $\underline{e}$ and $\overline{e}$ are drawn by dotted and dashed-dot lines, respectively.

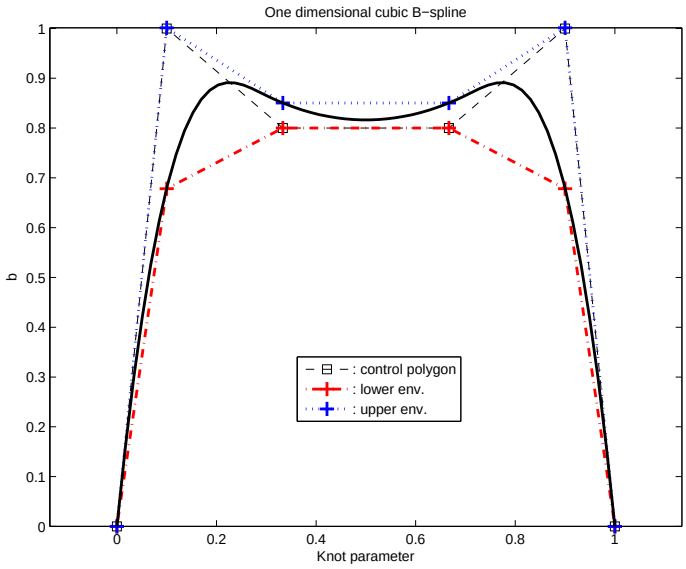


Figure 22: One dimensional cubic B-spline bounding envelopes.

3.2.2 Tight envelope for planar B-spline curves

A two dimensional planar B-spline curve $\boldsymbol{b}(u) = [b^1(u)\ b^2(u)]^\mathsf{T}$ is expressed in terms of the B-spline basis functions,

$$\boldsymbol{b}(u) = \sum_{j=0}^{m} \boldsymbol{b}_j N_j^d(u), \tag{39}$$

where, $\boldsymbol{b}_j = [b_j^1\ b_j^2]^\mathsf{T}$ are the control points. It is also assumed that the B-spline curve is clamped at the first and last control points by assigning the $(d+1)$ multiple knots at each the first and last knots.

At each Greville abscissa u_k^*, the one-dimensional bound by Eq. (38) constitutes a two-dimensional bounding box, whose ith axis is determined by the one-dimensional envelope as

$$\underline{e}^i(u_k^*) \le b^i(u_k^*) \le \overline{e}^i(u_k^*) \qquad i = 1, 2. \tag{40}$$

Let this axis-aligned bounding box be denoted by S^k, then the curve segment $\boldsymbol{b}(u)$, $u \in [u_k^*, u_{k+1}^*]$, lies in a convex combination of S^k and the consecutive box S^{k+1} owing to the linearity of $\underline{e}$ and $\overline{e}$, which is denoted by

$$H^k = \mathcal{L}(S^k,\ S^{k+1}). \tag{41}$$

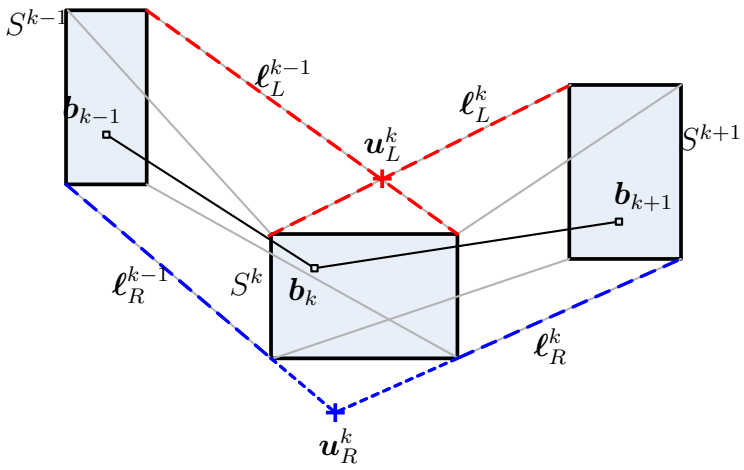


Figure 23: Constructing the envelope of a planar curve from neighboring bounding boxes.

Note that H_k is rendered as a convex hull of S^k and S^{k+1}, which is circumscribed by the edges of S^k and S^{k+1}, and lines connecting the corners of S^k and S^{k+1}. Let $\boldsymbol{v}_i^k$, $i = 1, \cdots, 4$, be the line segments connecting corresponding corners of S^k and S^{k+1}. Hence, $\boldsymbol{v}_1^k$ connects the lower left corner of S^k to the lower left corner of S^{k+1}, $\boldsymbol{v}_2^k$ connects the lower right corner of S^k to the lower right corner of S^{k+1}, and so on. Figure 23 shows these line segments. As mentioned above, the convex hull H^k over the knot $u \in [u_k^*, u_{k+1}^*]$ consists of parts of the edges of S^k and S^{k+1} and exactly two extra line segments $\boldsymbol{\ell}_L^k$ and $\boldsymbol{\ell}_R^k$ chosen among $\boldsymbol{v}_i^k$. The line segments $\boldsymbol{\ell}_L^k$ and $\boldsymbol{\ell}_R^k$ are separated by the line that connects the control points $\boldsymbol{b}_k$ and $\boldsymbol{b}_{k+1}$, thus $\boldsymbol{\ell}_L^k$ is denoted a left envelope line segment and $\boldsymbol{\ell}_R^k$ is a right envelope line segment. These line segments $\boldsymbol{\ell}_L^k$ and $\boldsymbol{\ell}_R^k$, $k = 0, \cdots, m$ are joined together to form piecewise linear envelopes of the B-spline curve $\boldsymbol{e}_L$ and $\boldsymbol{e}_R$, respectively. It might be the case, however, where two line segments do not intersect each other such as the line segments $\boldsymbol{\ell}_R^{k-1}$ and $\boldsymbol{\ell}_R^k$ in Fig. 23. In order to form piecewise linear envelopes by a set of line segments, those line segments are extended to find the intersection point $\boldsymbol{u}_R^k$. Consequently, the envelopes $\boldsymbol{e}_L$ and $\boldsymbol{e}_R$ are determined by a set of line segments between the intersection points $\boldsymbol{u}_L^k$ and $\boldsymbol{u}_R^k$. Figure 24 shows an example of two-dimensional bounding envelopes of

the given B-spline curve, which reveals that the entire B-spline curve stays inside the envelopes of $\boldsymbol{e}_L$ and $\boldsymbol{e}_R$.

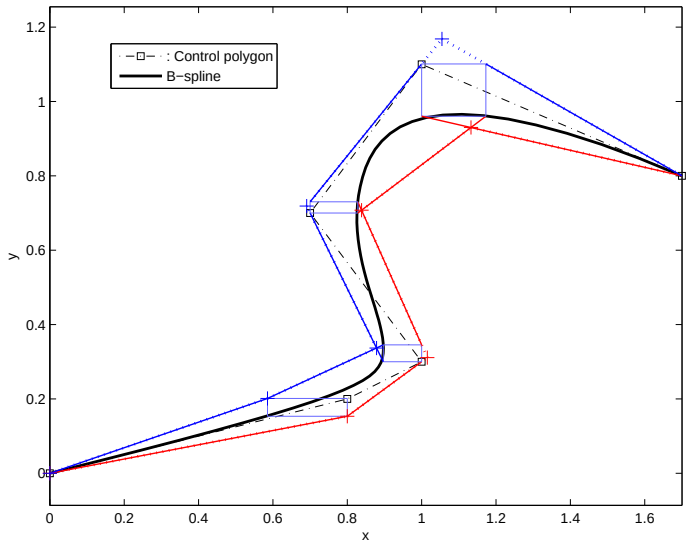


Figure 24: Bounding envelopes $\boldsymbol{e}_L$ and $\boldsymbol{e}_R$ of two-dimensional cubic B-spline .

3.3 Obstacle avoidance path optimization

3.3.1 Channel constraints for obstacle avoidance

We formulate an optimization problem to calculate a smooth curve which avoids a prescribed obstacle region. We adopt a B-spline curve, due to the inherent smoothness property of B-spline, as a reference path to the UAV. Hence, the path given by a B-spline curve is rendered a flyable path by the agent. In particular, we let the path be placed inside a feasible channel, thus avoiding the obstacle region. The channel separates the obstacle and feasible regions by two distinct polygonal lines to yield geometric constraints for optimization problem.

In Ref. [87], linear inequality constraints are incorporated in the B-spline function optimization. Given input channel polygon, the inequality expressions are formulated in conjunction with the lower and upper envelopes $\underline{e}$ and $\overline{e}$ of the B-spline function such that the B-spline function should stay between the polygons. On the other hand, because we deal with a planar B-spline curve in this research, the channel constraints

are formulated as geometric constraints. Hence, the given geometric channel should contain the envelopes of the B-spline curve.

To this end, we first introduce a signed distance-map function $f(\boldsymbol{x};\boldsymbol{\ell})$, $\boldsymbol{x} \in \mathbb{R}^2$ with respect to a polygonal line $\boldsymbol{\ell}$ to provide a metric for geometric constraints of the channel problem as follows,

$$f(\boldsymbol{x};\boldsymbol{\ell}) \triangleq s \cdot \min\{d_1,\ d_2\}, \tag{42}$$

where, $d_1 \in \mathcal{D}_1 = \{\|\boldsymbol{x} - \boldsymbol{c}_i\|,\ i = 0, \cdots, q\}$ is the distance from $\boldsymbol{x}$ to a corner point $\boldsymbol{c}_i$ of the polygonal line $\boldsymbol{\ell}_i$, $d_2 \in \mathcal{D}_2 = \{d(\boldsymbol{x}, \boldsymbol{\ell}_i),\ i = 0, \cdots, q-1\}$ is the perpendicular distance from $\boldsymbol{x}$ to the line segment $\boldsymbol{\ell}_i$ that connects two consecutive corner points $\boldsymbol{c}_i$ and $\boldsymbol{c}_{i+1}$, and s is a sign value which decides the location of the point $\boldsymbol{x}$ with respect to $\boldsymbol{\ell}$ as follows,

$$s = \begin{cases} +1, & \boldsymbol{x} \in \mathcal{O}, \\ 0, & \boldsymbol{x} \in \boldsymbol{\ell}, \\ -1, & \boldsymbol{x} \notin \mathcal{O}. \end{cases} \tag{43}$$

Figure 25 shows this distance-map function value with respect to the given zig-zag shape polygonal line. Far-away points from the line have bigger values, whereas points close to the line yield smaller values. The information about the relative location of the points with respect to the polygonal line is determined by its sign.

In order to formulate the inequality constraints as similar to those in Ref. [87], first of all, it follows from Fig 23 and accompanying discussions that the envelopes of a planar B-spline curve are characterized by the feature points $\boldsymbol{u}_L^k$ and $\boldsymbol{u}_R^k$ of the envelopes $\boldsymbol{e}_L$ and $\boldsymbol{e}_R$, respectively. Hence, if all these points are placed inside the feasible region, then each bounding box at $u = u_k^*$ of the B-spline curve will be completely contained in the feasible channel. Let $\boldsymbol{\ell}_L$ and $\boldsymbol{\ell}_R$ be the polygonal lines representing the obstacle boundaries, then the feature points $\boldsymbol{u}_L^k$ and $\boldsymbol{u}_R^k$ at each

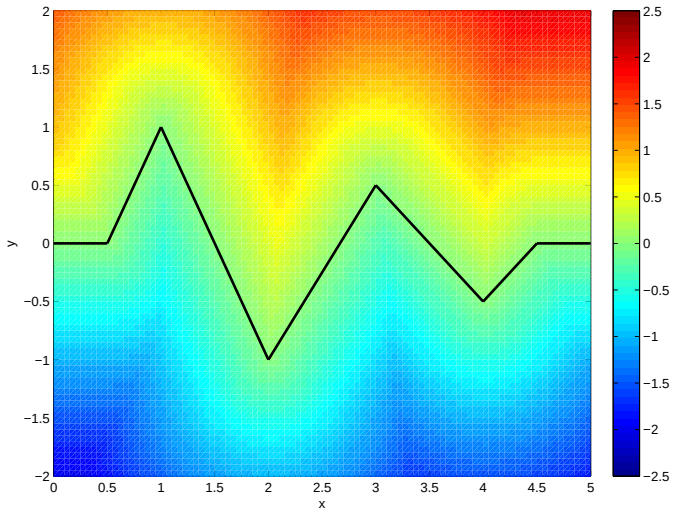


Figure 25: Signed distance map for an arbitrary polygonal line. The feasible region is characterized by the negative function values.

Greville abscissa $u = u_k^*$ should satisfy the following inequality relations,

$$f(\boldsymbol{u}_L^k; \boldsymbol{\ell}_L) \leq 0, \tag{44a}$$

$$f(\boldsymbol{u}_R^k; \boldsymbol{\ell}_R) \leq 0, \tag{44b}$$

where $k = 0, \cdots, m$.

Meanwhile, recall the fact that the envelopes of the B-spline curve are determined by the convex hull of each bounding box. Then the concave corner points of the obstacle boundaries, as marked by triangles in Fig. 26, should be excluded from the envelopes of the B-spline curve. To this end, we formulate inequality expressions utilizing the distance-map function in conjunction with the concave corner points and the piecewise linear envelopes as follow,

$$f(\boldsymbol{c}_l^{L_v}; \boldsymbol{e}_L) \geq 0 \tag{45a}$$

$$f(\boldsymbol{c}_m^{R_v}; \boldsymbol{e}_R) \geq 0 \tag{45b}$$

where, $l = 1, \cdots, n_{L_v}$ and n_{L_v} is the number of concave corner points of $\boldsymbol{\ell}_L$. Similarly, $m = 1, \cdots, n_{R_v}$ and n_{R_v} is the number of concave corner points of $\boldsymbol{\ell}_R$. Note that the

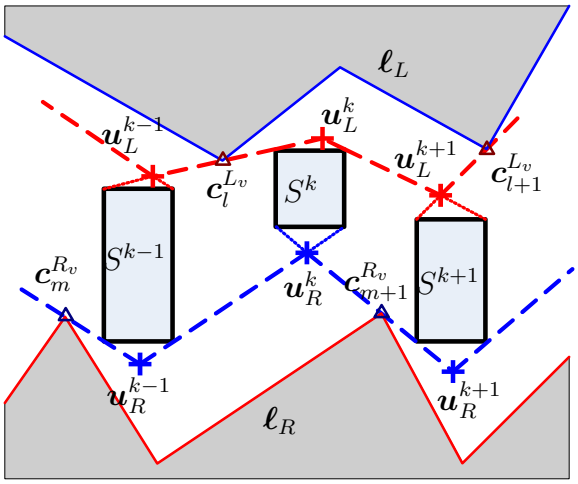


Figure 26: Geometric constraints formulation. The channel is given by two polylines $\boldsymbol{\ell}_L$ and $\boldsymbol{\ell}_R$, the envelope of the B-spline is drawn by the dashed lines, which is supposed to stay inside the channel.

positive condition in Eqs. (45) implies that the corner points are located outside the boundary envelopes of the B-spline curve. Consequently, the inequality constraints in Eqs. (44) and (45) ensure that the envelope of the B-spline stays between the channel, as depicted in Fig. 26.

3.3.2 Smooth curve optimization

We consider designing a smooth curve using a quartic B-spline, whose basis functions are computed from Eq. (31b) as degree 4 polynomials in terms of the knot parameter u. Hence, the quartic B-spline basis function preserves its continuity up to the third order derivative, thus resulting in the continuity of the derivative of curvature $\mathrm{d}/\mathrm{d}u(\kappa)$. Without loss of generality, the knot parameter is selected $u \in [0, 1]$, and the first and last knots have multiplicity 5 such as $u_0 = \cdots = u_4 = 0$ and $u_{m+1} = \cdots = u_{m+5} = 1$. Accordingly, the B-spline curve will be clamped at, or pass through, the first and last control points.

For the optimization, we manipulate $(m + 1)$ control points of the B-spline as $\boldsymbol{b}_j = [b_j^1 \; b_j^2]^\mathsf{T}$ $(j = 0, \cdots, m)$, which have direct influence on the shape of the curve. The number of control points is chosen along the complexity of the curve shape. In

general, the curve shape is closely related to the given channel geometry, subsequently the number of control points can be opted for the minimum number required to get a B-spline curve inside the channel. The knot sequence is initially given arbitrary non-decreading numbers in $(0, 1)$ by taking into account the number of control points, then can be altered with knot insertion if the envelopes of the B-spline curve need to be refined [87].

Two different performance indices are adopted to compute curves for attaining distinct optimization goals. In order to keep the B-spline curve as close as possible to a straight line, for which $\Delta_2 \boldsymbol{b}_j = 0$, we employ the cost function

$$\mathcal{J}_1 = \sum_{i=1}^{m-1} (\Delta_2 \boldsymbol{b}_j)^\mathsf{T} (\Delta_2 \boldsymbol{b}_j), \tag{46}$$

which implicitly minimizes the curvature variation of B-spline curve, thus resulting in a smooth B-spline curve. On the other hand, we suppose that the arc length of the B-spline curve is approximately captured by the total length of the control polygon $\boldsymbol{\ell}$. Hence, for the shortest path, we employ the cost function as the sum of the length of the piecewise control polygon,

$$\mathcal{J}_2 = \sum_{i=0}^{m-1} \|\boldsymbol{\ell}_i\|_2. \tag{47}$$

The constraints for the optimization problem is comprised of both equality constraints and inequality constraints. The equality constraints stipulate boundary conditions for position, heading, and curvature at each end point at $u_0 = 0$ and $u_{m+5} = 1$ as follows,

$$\boldsymbol{b}(0) = \boldsymbol{p}_0 \ , \qquad \boldsymbol{b}(1) = \boldsymbol{p}_f, \tag{48a}$$

$$\psi(0) = \psi_0 \ , \qquad \psi(1) = \psi_f, \tag{48b}$$

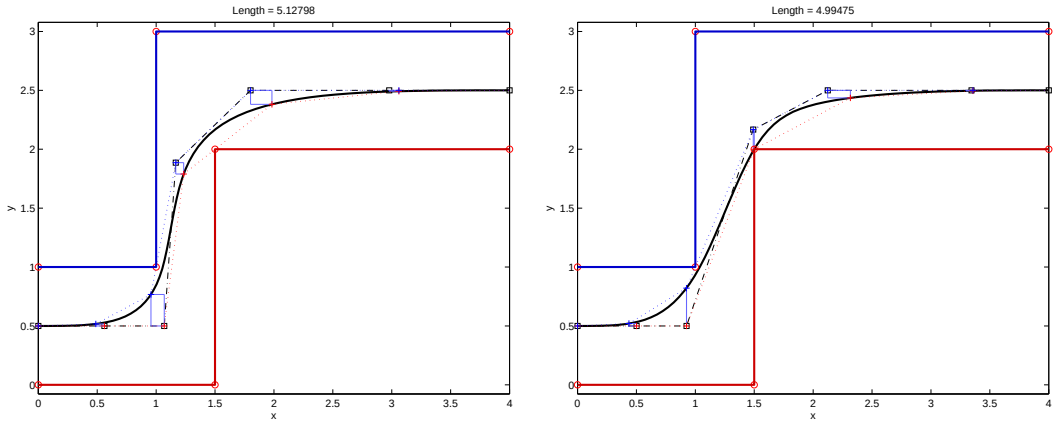
$$\kappa(0) = \kappa_0 \ , \qquad \kappa(1) = \kappa_f, \tag{48c}$$

where $\psi(u)$ and $\kappa(u)$ are the heading and the curvature of the B-spline curve at each

knot parameter. Inequality constraints are obtained from Eqs. (44) and (45) as the channel constraints.

The path optimization problem is formulated as follows. Given a knot sequence $\{u_k\}$, two polygonal lines for channel geometry, and boundary conditions for each end point, find a B-spline curve which minimizes the cost function in Eqs. (46) or (47) subject to the equality constraints in Eqs. (48) and the inequality constraints in Eqs. (44) and (45).

Figure 27(a) shows the optimization result using the cost function in Eq. (46). The constructed quartic B-spline curve is drawn by a solid line, and the envelopes are drawn by dashed lines. The B-spline as well as the envelopes stay inside the specified channel polygon. For the case of the shortest path using the cost function in Eq. (47), Fig. 27(b) reveals that the computed B-spline curve is rendered shorter than the previous case.



(a) A smooth curve in the channel (b) A shortest curve in the channel

Figure 27: Two optimization results.

3.4 Construct path templates for different channels

In this section we propose to construct path templates to be utilized for on-line path smoothing. The templates contain a set of planar B-spline curves which are then used for local path segments to avoid obstacles. An obstacle-free discrete path

sequence is provided by a high-level path planner [59], which constructs a channel as the obstacle-free region delimited by polygonal lines. Taking into consideration all possible combinations of path sequences, we solve a number of path optimization problems subject to different channel constraints, computing a set of quartic B-splines. The path templates are used for on-line path smoothing in conjunction with the high-level path planning algorithm in order to generate a smooth path which avoids obstacles.

3.4.1 Path rules within a finite horizon

Suppose a world environment $\mathcal{W} \subset \mathbb{R}^2$ is decomposed into a uniform cell decomposition which consists of square cells $c_{k,\ell}$ of dimension $1/2^J \times 1/2^J$ at resolution level J. We adopt the four-connectivity between cells, hence the high-level path planner computes an optimal path as a sequence of cells from the current cell to the goal cell. The path sequence is written in a *path word* by which the transitions toward the North, South, East, and West directions between two cells can be encoded by `N`, `S`, `E`, and `W`, respectively. We specify the range of interest within four-cell horizon from the current cell, as shown in Fig. 28. If the goal cell is located outside the horizon, then an optimal path sequence is computed in a such manner that the path eventually passes through one of the cells at the horizon boundary. Let a local path sequence from the current cell to reach the boundary cells be a *local path instance*. If a path is supposed to visit each cell only once, the number of possible combinations for local path instance turns out to be finite. In addition, taking advantage of the symmetry about the x-axis (East direction) and y-axis (North direction), we investigate the local path instances restricted on one quadrant of the 7×7 cell grid.

Without loss of generality, we consider the local path instances on the first quadrant, as shown in Fig. 29. From the square cell geometry, it follows that we can also take advantage of the symmetry about the diagonal axis. Hence, as shown in Fig. 29,

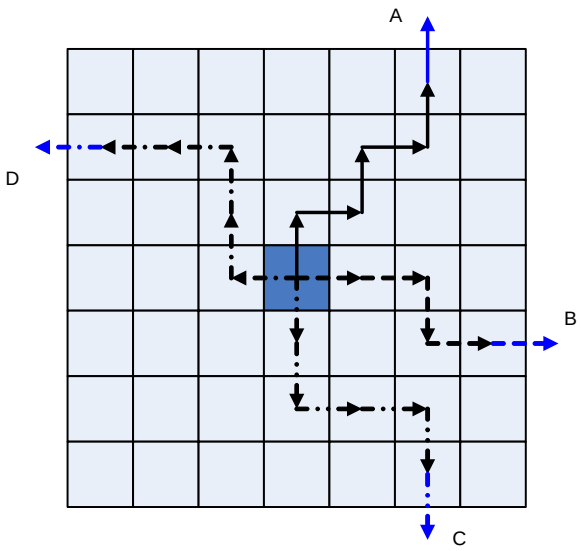


Figure 28: Examples of path sequences starting from the current cell at the center. We adopt the four-connectivity between cells. The goal cell is supposed to be located beyond the horizon. Possible path sequences are given as examples of the optimal path. The path A is written by `NENEN`$\cdots$, the path B is `EESE`$\cdots$, the path C is `SSEES`$\cdots$, and the path D is `WNNWW`$\cdots$.

two path instances of `ENNEN` and `NEENE` are similar to each other so that they are considered to be derived from the same local path instance.

Subsequently, we only investigate the local path instances from the current cell to one of the top boundary cells $c_{0,3}$, $c_{1,3}$, $c_{2,3}$, and $c_{3,3}$ shown in Fig. 29. Any local path instances can then be derived from these path instances by applying the symmetric operations along the horizontal, vertical, diagonal axes. Among all possible combinations of path sequences to reach the top boundary cells, we describe the necessary path rules to determine a unique local path instance, which are enlisted as follows,

1. *(Terminal conditions)* Suppose a local path instance is restricted inside the first quadrant, that is, it never goes outside the horizon before it reaches a terminal cell on the top boundary. The terminal cell should be one of the top boundary cells, except the cell $c_{3,3}$. The reason for this is attributed to the four-connectivity between cells; The path instance that reaches $c_{3,3}$ necessarily

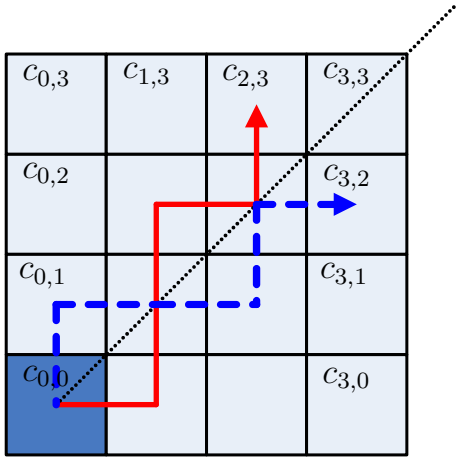


Figure 29: Local path instances on the first quadrant. From the additional symmetry about the diagonal axis, it is possible to transform the path instance drawn in dashed line (`NEENE`) to the path instance drawn in solid line (`ENNEN`). Path rules are given in order to determine unique path instances reaching the cell at the top boundary.

passes the adjacent boundary cell either $c_{2,3}$ or $c_{3,2}$, thus we regard the path instances to the cell $c_{3,3}$ as a subset of the path instances to $c_{2,3}$ and exclude from the consideration. In order to come up with a path sequence that reaches the terminal cell, the corresponding path word should have certain numbers of occurrence of `N`, `S`, `E`, and `W` satisfying the following conditions,

$$\sum \left(\#\mathtt{N} - \#\mathtt{S}\right) = 3,$$

$$\sum \left(\#\mathtt{E} - \#\mathtt{W}\right) = \begin{cases} 0 & \text{for } c_{0,3}, \\ 1 & \text{for } c_{1,3}, \\ 2 & \text{for } c_{2,3}. \end{cases}$$

2. *(Self-avoiding path)* The path must visit each cell exactly once, and never intersect itself. From this rule, we explicitly prevent a pathological case such as a cyclic loop from being considered in the templates. This type of path can be described by a self-avoiding walk [89] on a 4×4 cell grid. The total number of self-avoiding walks on an $m \times n$ grid, which starts from a corner and ends at the opposite corner by only horizontal and vertical steps, is computed using

Table 3: Number of self-avoiding walks on an $m \times n$ grid

m, n	2	3	4	5
2	2	4	8	16
3	4	12	38	125
4	8	38	184	976
5	16	125	976	8512

a recurrence relation [42]. Table 3 gives the first few numbers of such walks for small m and n. Similarly, the number of candidate of self-avoiding paths from the current cell $c_{0,0}$ to the top boundary cells is calculated from Table 3 utilizing recurrence relationship. Among these candidates, only certain number of self-avoiding path will be considered in the path templates.

3. *(Path optimality)* The high-level path planning algorithm is supposed to provide an optimal path sequence. The optimal path sequence is calculated as such that it minimizes the accumulated transition cost from the current node to the goal node. Typically, an directed edge cost is assigned to each transition of `N`, `S`, `E`, `W`, taking into account the cost associated with the cells as follows,

$$\mathcal{J}(u, v) = f(v) + \alpha g(u, v), \tag{49}$$

where, f is a positive obstacle cost associated with the target cell v, g is (Euclidean) distance cost between u and v, and $\alpha \geq 0$ is a weight constant. Consider now the path word `ENW` which represents the transition among four cells u, v, w, and z in order. The accumulated cost for this transition is computed by

$$\mathcal{J}(u, v) + \mathcal{J}(v, w) + \mathcal{J}(w, z) > \mathcal{J}(u, z), \tag{50}$$

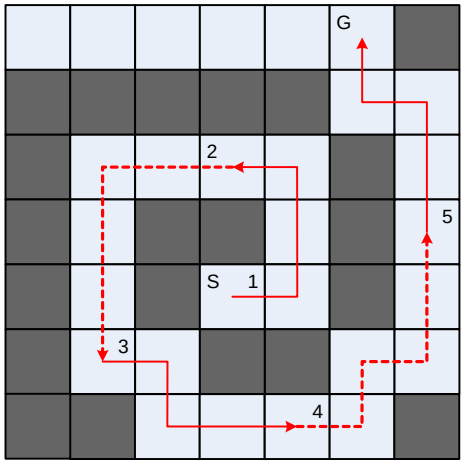
which turns out to be greater than the direct transition cost from u to z by a single path sequence `N`. It follows that the transition `ENW` is not an optimal, and neither of `ESW`, `NES`, `NWS`, etc. Consequently, we disregard any non-optimal path sequence when investigating the candidates of self-avoiding paths.

When establishing the previous path rules, we assumed that the local path instance necessarily ends up the top boundary cells, without escaping the quadrant. In certain cases, however, the path sequence might be rather provided in such way that it comes to cross the quadrants. In other words, the path sequence which starts from the current cell at the center of grid, is comprised of cells in more than two quadrants. If this is the case, we can infer that the path sequence will finally exit the finite horizon after passing through at least two quadrants, which makes it difficult to take advantage of the symmetry of the templates. In particular, the number of templates will increase, if one wants to consider all possibilities of inter-quadrant transitions, thus losing the benefit of using templates. In order to retain the symmetry of templates, we append additional cells between the quadrants to be considered as terminal cells other than the top boundary cells. Applying the path rules to a 7×7 cell grid (thus 4×4 cell grid for the first quadrant), only the cell $c_{0,2}$ (See Fig. 29) can be considered as an additional terminal cell. The other cells on the axis can not be terminal cells since the local path instances reaching them would conflict with the path rules. Then any possible local path instances starting from the center cell to $c_{0,2}$, certainly satisfying the path rules, are appended to the path templates.

The path templates for local path instances on the first quadrant are summarized in Table 4. Figure 30 shows an example of utilizing the path templates on a given path sequence. The starting cell is located in the middle area, where the path sequence is computed avoiding the shaded obstacle cells. In order to reach the goal cell, five local path instances are required as they are connected one another at one cell. Since each local path instance is written in path word, the corresponding path templates are incorporated with required symmetry operations, which are horizontal(H), vertical(V), diagonal(D) reflections, in order to adopt the templates to the actual path words.

Table 4: Path templates for local path instances on the first quadrant.

Destination cell	Path words			
$c_{0,3}$	NNN	ENNWN	EENNWWN	
$c_{1,3}$	NNEN	NENN	ENNN	EENNWN
$c_{2,3}$	NNEEN	NENEN	ENNEN	
	NEENN	ENENN	EENNN	
$c_{0,2}$	ENNW	EENNWW		



Path #	Path words	Template	Operations
1	ENNW	ENNW	-
2	WWSSS	EENNN	H, V
3	ESEE	NENN	H, D
4	ENENN	ENENN	-
5	NNWN	NNEN	V

Figure 30: Example incorporating the path templates on a complex path sequence. Five local path instances are are connected each other to reach the goal cell. The actual path words are equivalently recovered from the path templates with corresponding symmetry operations, which are horizontal(H), vertical(V), diagonal(D) reflections.

3.4.2 Construct B-spline path templates

The B-spline path templates are composed of a set of B-spline curves which is supposed to be placed inside the channels. By assumption of the high resolution representation of $\mathcal{W}$, the cells of the optimal path sequence are regarded as a feasible region for the agent to fly safely. Hence, the shape of the channel is determined from the optimal path sequence by taking into account the square cell geometry, as the border lines around the cells of the local path instance are joined together to yield a channel polygon consisting of polylines (Left, Right) for the channel constraints. The boundary conditions of each B-spline curve are chosen, for the convenience sake,

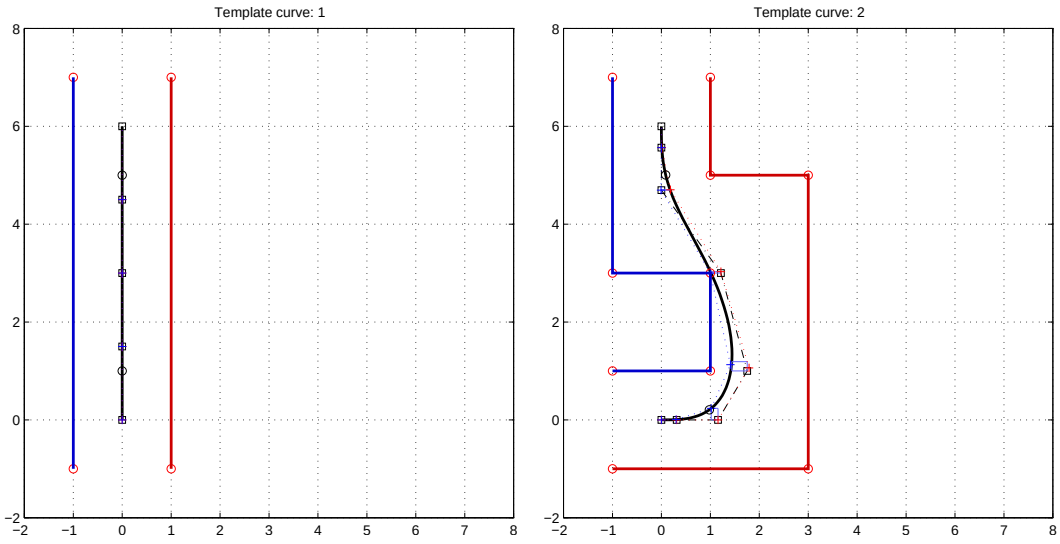
such that the B-spline curve starts from the center of the first cell of the local path instance, ends at the center of the last cell of the local path instance. The heading angles at each end of the curve is chosen such that the tangent vector at the point directs toward the center of the next adjacent cell, whereas the curvature values are set to be all zero. Hence, we manage to solve the optimization problem discussed in Section 3.3 for minimizing one of the cost in Eqs. (47) and (46), or combination of both. Figure 31 shows the results of optimization for path templates using B-splines corresponding each local path instance shown in Table 4.

3.5 On-line path smoothing algorithm

In this section, we present an on-line path smoothing algorithm incorporating the B-spline path templates. Given a discrete path sequence from a high-level path planner, each instance of the B-spline path templates becomes a part of the smooth path. Hence, the on-line path smoothing algorithm connects these path segments resulting in the smooth path over the entire section.

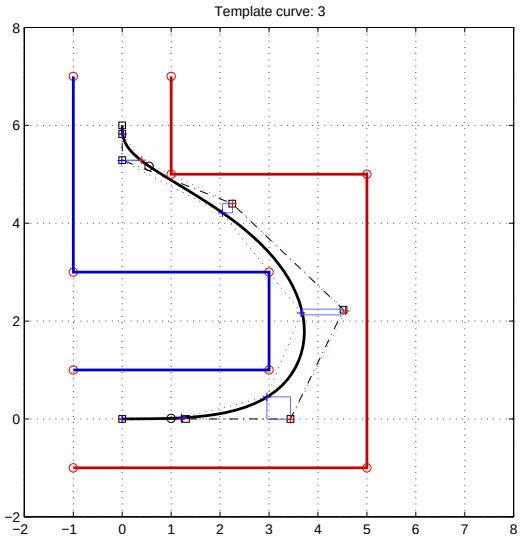
3.5.1 Stitching the path segments

Along with the earlier discussion, two B-spline curves which are chosen from the path templates meet each other at one junction point. Due to the different boundary conditions with respect to tangent direction at each end of the curves, it follows that a heading angle discontinuity occurs at the junction point. Hence, in order to get a smooth path segment over two consecutive B-spline curves, we should stitch them with a transient B-spline curve, hence preserving the continuity property over distinct B-spline curves. To this end, we let $\boldsymbol{p}_a$ and $\boldsymbol{p}_b$ be the points on the leading and the following B-spline curves, respectively, other than the end points. Suppose the transient curve intersects these points at its own end points, then it follows that preserving continuity condition over distinct B-spline curves is assured by imposing the continuity conditions at these points. Hence, the transient curve, which is supposed

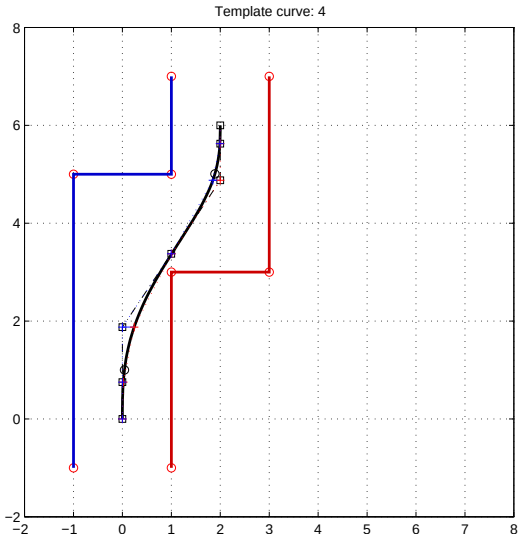


(a) NNN

(b) ENNWN

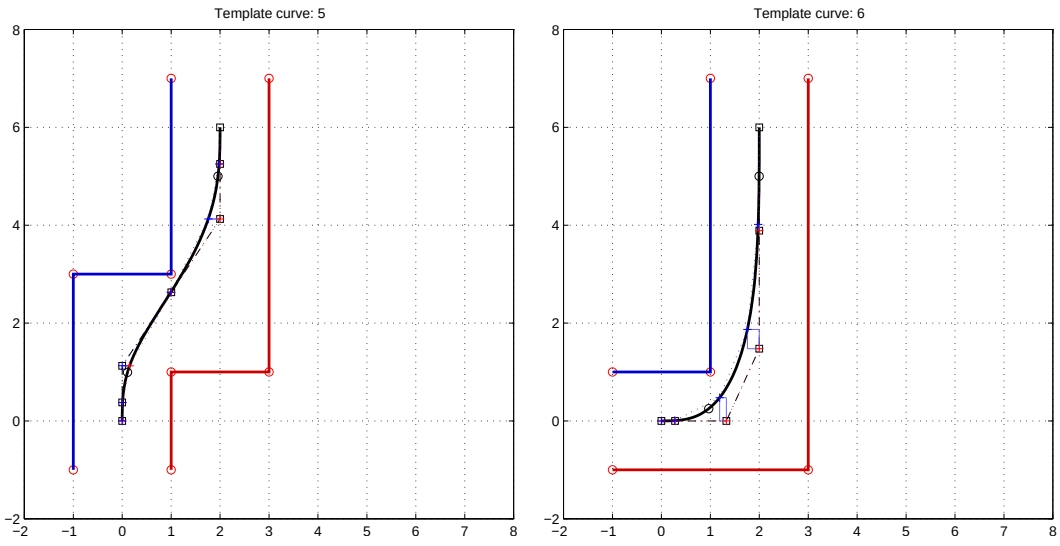


(c) EENNWWN



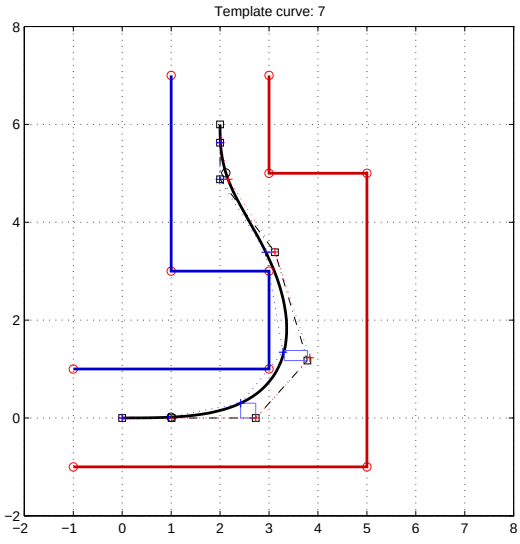
(d) NNEN

Figure 31: Path templates results from the path optimization using B-spline curves. Each plot corresponds to the local path instance in Table 4.

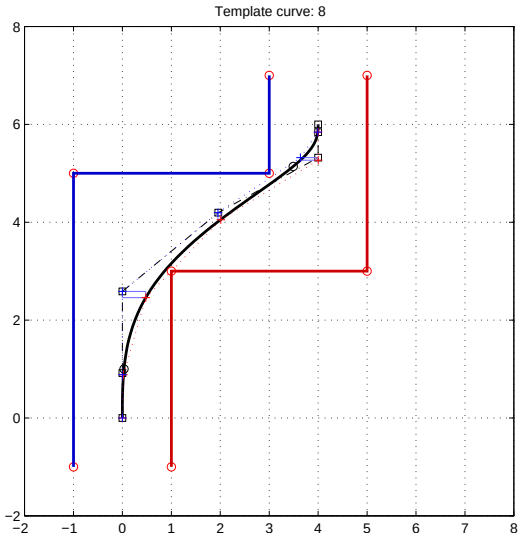


(e) NENN

(f) ENNN

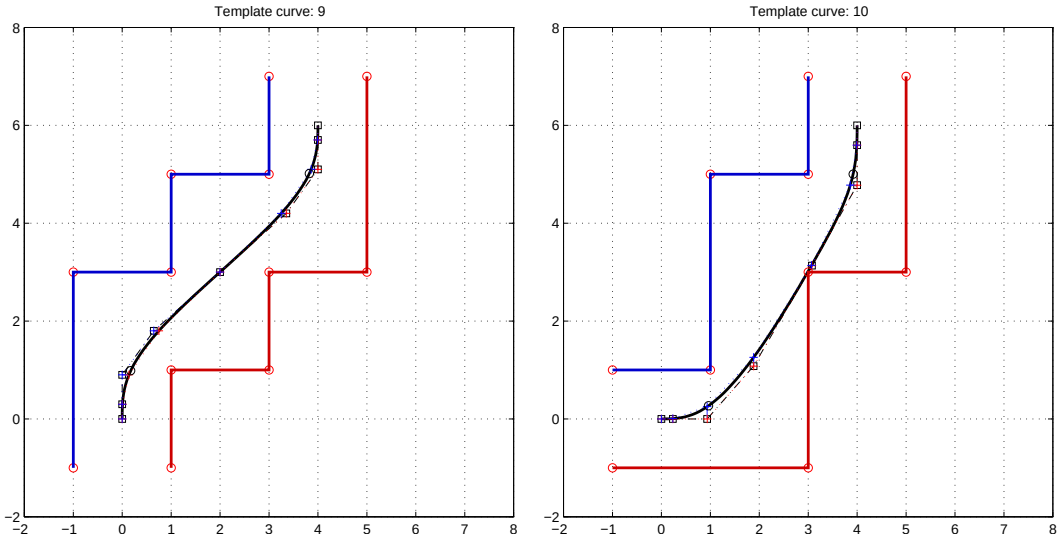


(g) EENNWN



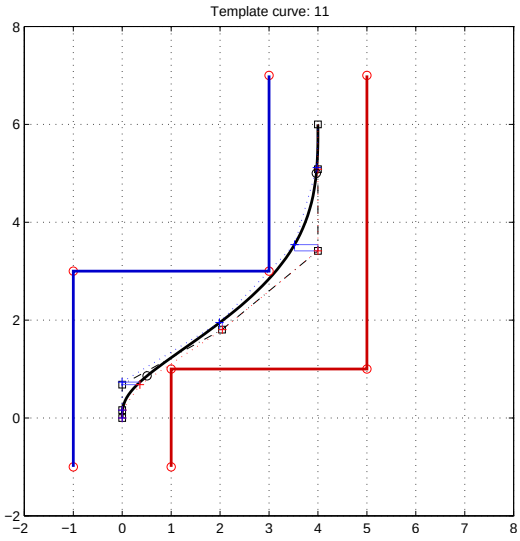
(h) NNEEN

Figure 31: (Continued) Path templates results from the path optimization using B-spline curves. Each plot corresponds to the local path instance in Table 4.

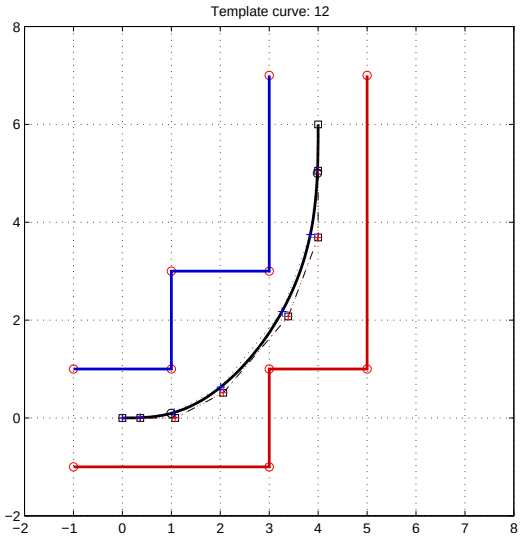


(i) NENEN

(j) ENNEN

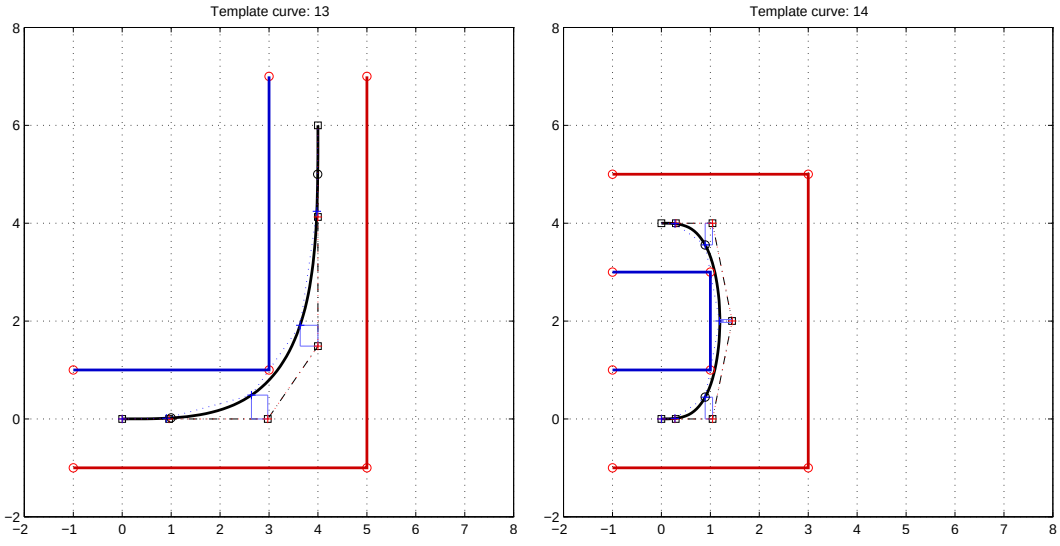


(k) NEENN

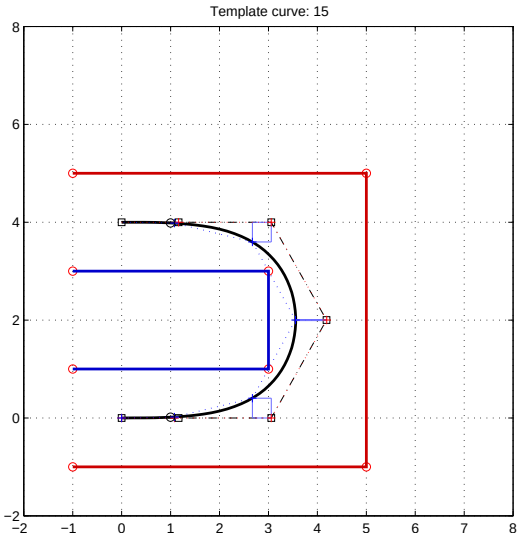


(l) ENENN

Figure 31: (Continued) Path templates results from the path optimization using B-spline curves. Each plot corresponds to the local path instance in Table 4.



(m) `EENNN` (n) `ENNW`



(o) `EENNWW`

Figure 31: (Continued) Path templates results from the path optimization using B-spline curves. Each plot corresponds to the local path instance in Table 4.

to stitch two distinct B-spline curves, satisfies the continuity conditions not only for position but also for heading angle and curvature value at each end.

The minimum order of the B-spline curve that satisfies the continuity conditions described above is four. Subsequently, we employ a transient cubic B-spline curve which is defined on a fixed clamped knot vector, $u = [0, 0, 0, 0, 1/3, 2/3, 1, 1, 1, 1]$. Hence, the unknown parameters for determining this transient B-spline curve are six control points, among which the first three control points are related to the boundary conditions associated with the leading B-spline curve, and the rest are related to the boundary conditions associated with the following B-spline curve. Suppose the boundary conditions at the intersection point with the leading B-spline curve are given as follows. The intersection point is specified by $\boldsymbol{p}_a = [x_a, y_a]$, the heading angle at $\boldsymbol{p}_a$ is given ψ_a, and the curvature value is given κ_a. Let the control points be given $\boldsymbol{p}_i = [x_i, y_i]$, $i = 0, \cdots, 5$, on which we impose the following boundary conditions,

$$x_a = x_0, \tag{51a}$$

$$y_a = y_0, \tag{51b}$$

$$\psi_a = \tan^{-1}\left(\frac{y_0'}{x_0'}\right). \tag{51c}$$

where, x_0' and y_0' are the derivative with respect to the knot parameter at the control point $\boldsymbol{p}_0$. Note that, with the adopted clamped knot vector above, the cubic B-spline basis functions evaluated at the knot $u = 0$ are computed as follows,

	b_0	b_1	b_2	b_3	$\cdots$
$N_i^3(0)$	1	0	0	0	
$N_i^{3\prime}(0)$	-9	9	0	0	$\cdots$
$N_i^{3\prime\prime}(0)$	54	-81	27	0	

(52)

Hence, the first derivative at $\boldsymbol{p}_0$ is evaluated with the knot parameter $u = 0$ as follows,

$$x_0' = -9x_0 + 9x_1, \tag{53a}$$

$$y_0' = -9y_0 + 9y_1. \tag{53b}$$

From Eqs. (51c) and (53), it follows that

$$x_1 = x_a + R_a \cos\psi_a, \tag{54a}$$

$$y_1 = y_a + R_a \sin\psi_a, \tag{54b}$$

where R_a is the distance between $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$ as a design parameter. In addition, we impose the curvature boundary condition as follows

$$\kappa_a = \frac{x_0{}' y_0{}'' - y_0{}' x_0{}''}{\left(x_0{}'^2 + y_0{}'^2\right)^{3/2}}, \tag{55}$$

where, $x_0{}''$ and $y_0{}''$ are the second derivative with respect to the knot parameter at $\boldsymbol{p}_0$. Using Eqs. (52) and (53) after algebraic manipulation results in the following expression regarding x_2 and y_2,

$$-x_2 \sin\psi_a + y_2 \cos\psi_a = 3R_a^2\kappa_a + y_a \cos\psi_a - x_a \sin\psi_a. \tag{56}$$

In order to solve for x_2 and y_2, we adopt an additional equation in terms of x_2 and y_2 which determines a unique solution of x_2 and y_2 as follows. Suppose $\boldsymbol{p}_2^{\perp}$ is the point at the distance $2R_a$ along the line $\overline{\boldsymbol{p}_0\boldsymbol{p}_1}$ from $\boldsymbol{p}_0$ (See Fig. 32). The control point $\boldsymbol{p}_2$ can then be chosen uniquely by imposing the additional condition such that the projection of $\boldsymbol{p}_2$ onto the line $\overline{\boldsymbol{p}_0\boldsymbol{p}_1}$ ends up with the design point $\boldsymbol{p}_2^{\perp}$. It follows that

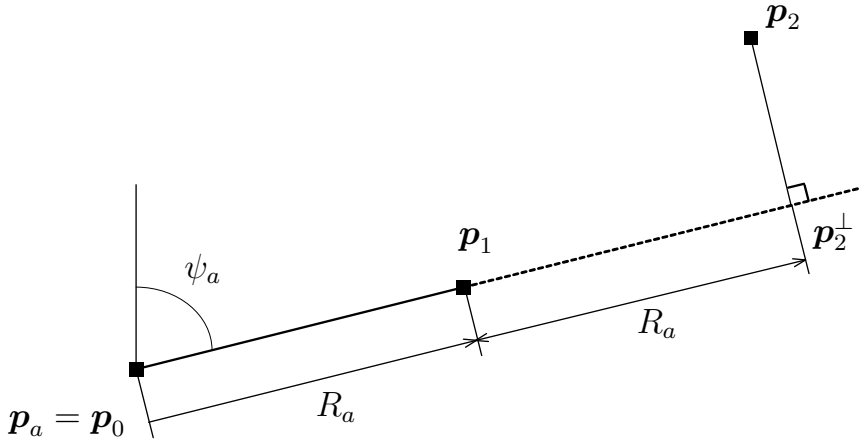


Figure 32: Determine the unique $\boldsymbol{p}_2$ in terms of $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$ in conjunction with the design parameter R_a.

with extra algebraic manipulation, we obtain the additional equation as follow,

$$x_2 \cos\psi_a + y_2 \sin\psi_a = 2R + x_a \cos\psi_a + y_a \psi_a. \tag{57}$$

In a similar manner, we impose the boundary conditions at the end of the transient B-spline as follows. With the cubic B-spline basis functions evaluated at the knot $u = 1$,

$$\begin{array}{c|ccccc} & \cdots & x_2 & x_3 & x_4 & x_5 \\ \hline N_i^3(1) & & 0 & 0 & 0 & 1 \\ N_i^{3\prime}(1) & \cdots & 0 & 0 & -9 & 9 \\ N_i^{3\prime\prime}(1) & & 0 & 27 & -81 & 54 \end{array} \tag{58}$$

and the given boundary conditions such that the intersection point is specified by $\boldsymbol{p}_b = [x_b, y_b]$, the heading angle at $\boldsymbol{p}_b$ is given ψ_b, and the curvature value is given κ_b, we obtain the following formulas to determine the three control points,

$$x_5 = x_b, \tag{59a}$$

$$y_5 = y_b, \tag{59b}$$

$$x_4 = x_b - R_b \cos\psi_b, \tag{59c}$$

$$y_1 = y_b - R_b \sin\psi_b, \tag{59d}$$

$$-x_3 \sin\psi_b + y_3 \cos\psi_b = 3R_b^2 \kappa_b + y_b \cos\psi_b - x_b \sin\psi_b, \tag{59e}$$

$$x_3 \cos\psi_b + y_3 \sin\psi_b = -2R_b + x_b \cos\psi_b + y_b \sin\psi_b. \tag{59f}$$

Figure 33 shows an example of stitching the two B-spline curve derived from the path templates by a transient cubic B-spline.

3.5.2 Simulation results of the on-line path smoothing algorithm

In this section we present simulation results of the on-line path smoothing in conjunction with the $\mathcal{D}^*$-lite path planning algorithm. It is assumed that the agent navigates over the unknown environment, while updating the map with the information gathered from a proximity sensor. The world data is given by 256$\times$ 256 pixels. We adopt

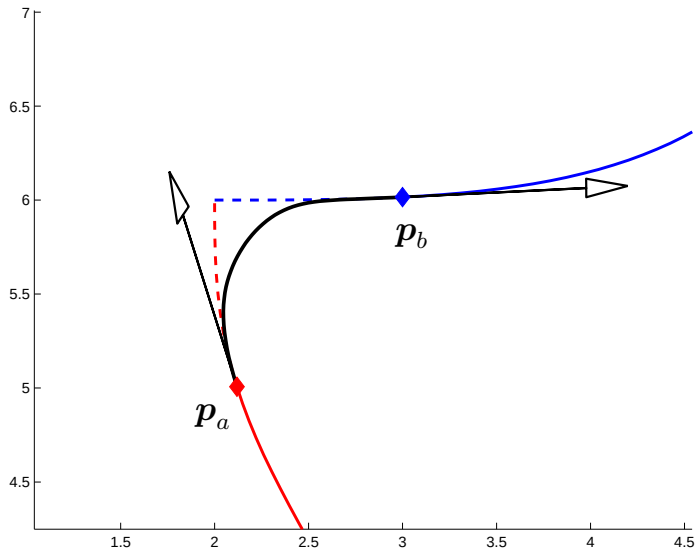


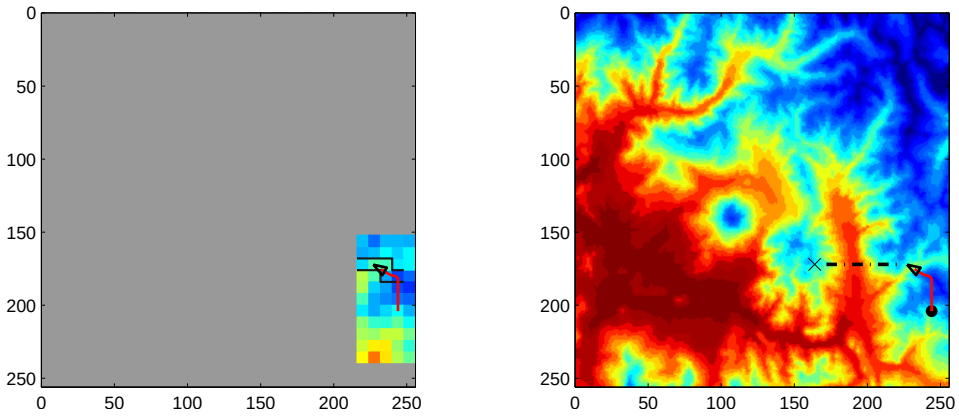
Figure 33: Example of stitching the two B-spline curve with a cubic B-spline curve

a uniform cell decomposition of cell size 8×8 pixels at the level $J = 5$. The range of the proximity sensor is chosen to be $r_5 = 28$, thus resulting in the high resolution window by a 7×7 square cell grids.

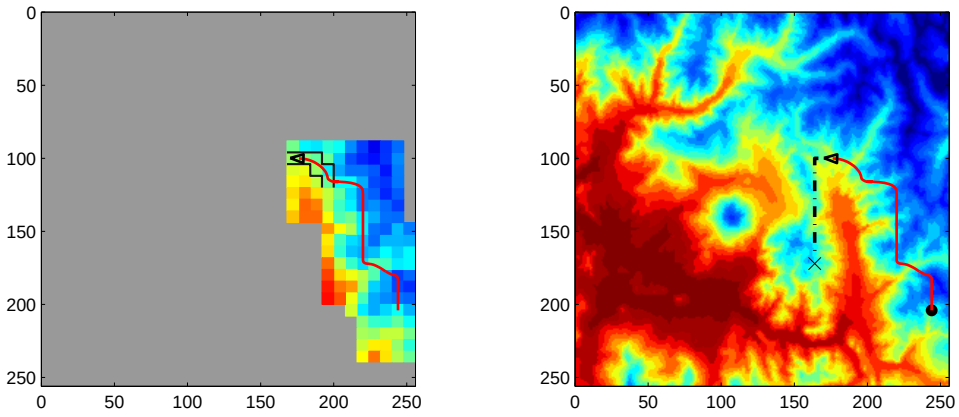
The results from the on-line path smoothing algorithm with the $\mathcal{D}^*$-lite path planning algorithm are shown in Fig. 34. Specifically, Fig. 34 shows the evolution of the path at different time steps as the agent moves to the final destination. At each step, the best proposed path is drawn by a dashed-dot line and the actual path followed by the agent is drawn by a solid line. In each step a channel is drawn by thin polylines that correspond to the discrete path sequence. The actual path is constructed by joining the smooth path segments derived from the B-spline path templates. The $\mathcal{D}^*$-lite algorithm updates occur whenever the agent approaches closed to the end of each path segment, giving a (possibly new) path sequence. Subsequently, the previous curve and the newly derived B-spline curve are stitched with a transient B-spline curve. This process is repeated until the UAV reaches the final destination, as shown in Fig. 34(c)

3.6 Summary

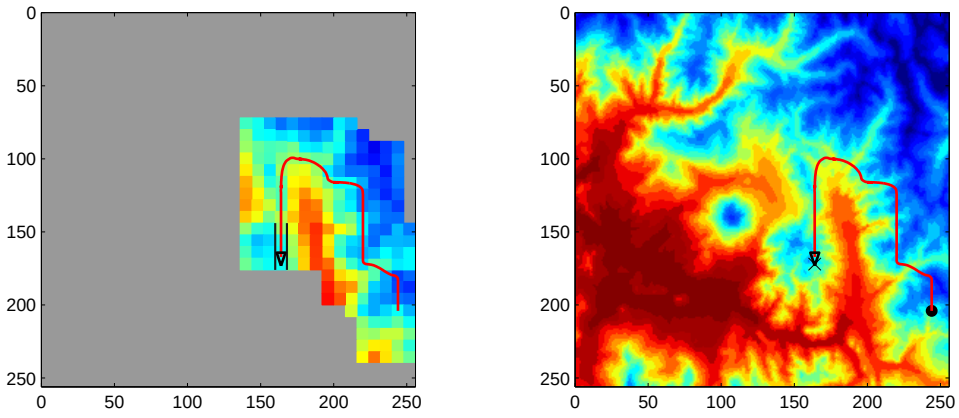
In this chapter, we presented the on-line path smoothing algorithm incorporating the path templates for generating a smooth path derived from the high-level path planner. The path templates are comprised of a set of B-spline curves, which have been obtained from the off-line optimization in the manner that each path instance stays inside the prescribed channel, hence the path efficiently avoids obstacles outside channels. In conjunction with the high-level path planning algorithm, the on-line implementation of the proposed algorithm involves finding the corresponding path segments and stitching them together while preserving the continuity of the curve. The simulation results with the $\mathcal{D}^*$-lite path planning algorithm validates the effectiveness of the proposed algorithm, having minimal on-line computational cost to get a smooth path for UAV to fly along.



(a) $t = t_{14}$



(b) $t = t_{30}$



(c) $t = t_f$

Figure 34: On-line path smoothing in conjunction with replanning using $\mathcal{D}^*$-lite algorithm. Dashed-dot lines represent the currently tentative optimal path obtained from the $\mathcal{D}^*$-lite algorithm, based on the distance cost outside the high resolution area. Actual path followed by the agent is derived from the B-spline path templates, which are represented by solid lines.

CHAPTER IV

PATH FOLLOWING CONTROL USING BACKSTEPPING AND PARAMETER ADAPTATION

4.1 Introduction

Modern UAVs are envisioned to replace human pilots in various missions. In order to accomplish these missions with minimal human intervention, the operation of a UAV should be fully automated from the top level of path planning, down to the inner control loop level. At the top level of the control hierarchy, the path planning algorithm computes a rough approximation of the optimal path toward the goal. A path generation algorithm then smooths the path yielding a dynamically feasible, flyable path, by taking into account the kinematic constraints of the UAV. Finally, the path following algorithm is responsible for guiding the UAV to stay close to the designed path.

Various control approaches in the literature have been proposed to address the path following problem: Niculescu [102] introduced a lateral track control law, Park et al. [106] proposed a simple, yet effective, nonlinear control logic and demonstrated it experimentally, Ren and Beard [116] considered the problem of constrained trajectory tracking, Nelson et al. [100] proposed a path following control using vector fields to guide the UAV on the desired path, and Rysdyk [120] proposed a guidance law based on the 'good helmsman' behavior.

Motivated by the method proposed in Ref. [96], kinematic control laws have been used to regulate the distance error from the reference path for unicycle-type mobile robots [75, 76]. The key aspect of the proposed algorithms in Refs. [75, 76] is that the control laws explicitly incorporate the controlled motion of a 'virtual target' to

be tracked along the path. In this chapter, we present a nonlinear path following algorithm, which is an extension of the one by [75] for UAV path following control. We apply a standard backstepping technique to compute the roll angle command from the heading rate command of the kinematic control law. A parameter adaptation technique is then proposed to compensate for the inaccurate time constant of the roll closed loop, yielding robust performance during controller implementation. The path following control algorithm is validated through a high-fidelity hardware-in-the-loop simulation (HILS) environment to show the applicability of the presented algorithm on the actual system.

4.2 Problem Description

A fixed-wing UAV is equipped with a low-level autopilot with on-board sensors that provides feedback control for attitude angles, air speed, and altitude. In a typical search mission, the air speed and the altitude are kept constant, so that the UAV stays inside the safe flying envelope. Suppose that the inertial speed (V) and the course angle (χ; inertial speed heading) are directly measured using an on-board GPS sensor. A simplified kinematic model in the two dimensional plane is given as follows,

$$\begin{aligned} \dot{x} &= V\cos\chi, \\ \dot{y} &= V\sin\chi, \\ \dot{\chi} &= \omega, \end{aligned} \tag{60}$$

where (x, y) denotes the inertial position, and ω is the heading rate of the UAV. By expressing the equations of motion in terms of the ground speed and the course angle, the effect of the wind velocity on the dynamics is removed. Furthermore, by using inertial measurements for path planning control, wind disturbance is naturally rejected so that the performance of path following controller can be improved significantly. For a fixed-wing UAV at a banked-turn maneuver with no sideslip, and assuming

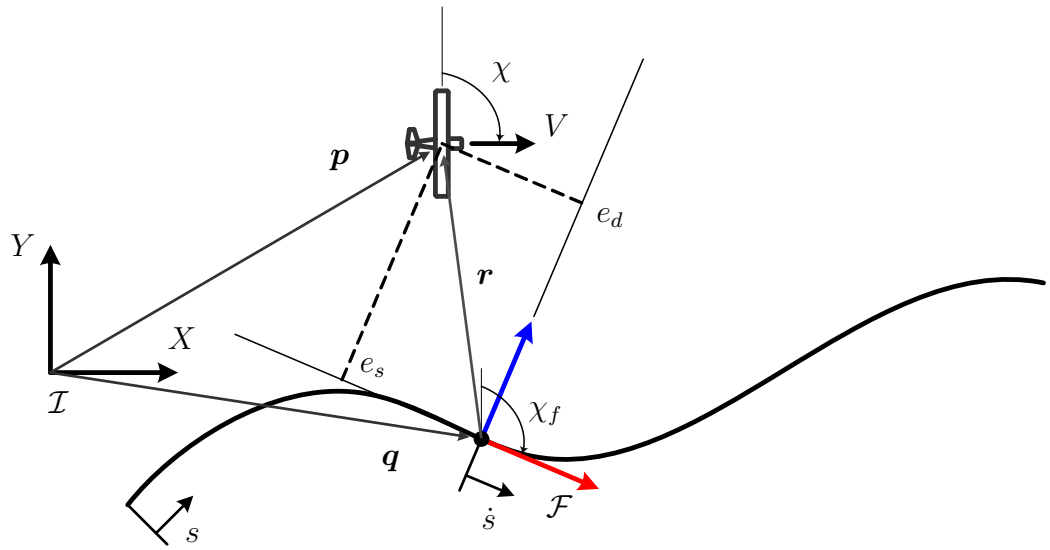


Figure 35: Definition of the Serret-Frenet frame for the path following problem.

that $|\phi| < \phi_{\max} < \pi/2$, the heading rate ω is induced by the roll angle ϕ with the inertial speed V as follows,

$$\omega = \frac{g}{V} \tan \phi, \tag{61}$$

where the roll angle is assumed to be controlled by an inner PID loop. Properly tuned PID gains result in closed-loop roll dynamics which resemble a first-order system,

$$\dot{\phi} = \frac{1}{\lambda_\phi}(\phi_c - \phi), \tag{62}$$

where ϕ_c is the roll angle command and $\lambda_\phi > 0$ is the time constant.

Consider now a UAV flying along a planar path as shown in Fig. 35. Denote the inertial position of the UAV by $\boldsymbol{p} = [x \ y]^\mathsf{T} \in \mathbb{R}^2$. The geometric path is defined in terms of the arc-length parameter s. For any given s, the inertial position of the point on the path associated with s is denoted by $\boldsymbol{q}(s) \in \mathbb{R}^2$, at which the Serret-Frenet frame is attached, and moves along the path with speed $\dot{s}$. The x-axis of the Serret-Frenet frame is aligned with the tangent vector to the path at $\boldsymbol{q}(s)$ and has an angle $\chi_f(s)$ with respect to the inertial frame $\mathcal{I}$. Let the error vector $\boldsymbol{e}$ in the Serret-Frenet frame be decomposed in the along-track error e_s and the cross-track error e_d. Then the inertial error vector $\boldsymbol{r} = \boldsymbol{p} - \boldsymbol{q}(s)$ expressed in the Serret-Frenet frame is obtained

by

$$\boldsymbol{e} = \mathbf{R}^{\mathsf{T}}(\chi_f)\big(\boldsymbol{p} - \boldsymbol{q}(s)\big), \tag{63}$$

where,

$$\mathbf{R}(\chi_f) = \begin{bmatrix} \cos\chi_f & -\sin\chi_f \\ \sin\chi_f & \cos\chi_f \end{bmatrix}, \tag{64}$$

is the rotation matrix from the Serret-Frenet frame to the inertial frame.

By differentiating Eq. (63) with respect to time, it follows that

$$\dot{\boldsymbol{e}} = \dot{\mathbf{R}}^{\mathsf{T}}(\chi_f)\big(\boldsymbol{p} - \boldsymbol{q}(s)\big) + \mathbf{R}^{\mathsf{T}}(\chi_f)\big(\dot{\boldsymbol{p}} - \dot{\boldsymbol{q}}(s)\big), \tag{65}$$

where,

$$\dot{\mathbf{R}}(\chi_f) = \mathbf{R}(\chi_f)\mathbf{S}(\dot{\chi}_f), \tag{66}$$

where, $\mathbf{S}(\dot{\chi}_f)$ is the skew-symmetric matrix,

$$\mathbf{S}(\dot{\chi}_f) = \begin{bmatrix} 0 & -\dot{\chi}_f \\ \dot{\chi}_f & 0 \end{bmatrix}. \tag{67}$$

It follows from Eq. (60) that

$$\dot{\boldsymbol{p}} = \mathbf{R}(\chi) \begin{bmatrix} V \\ 0 \end{bmatrix}. \tag{68}$$

Notice that $\dot{\boldsymbol{q}}$ is the time derivative of the point $\boldsymbol{q}(s)$, whose speed is represented by $\dot{s}$ when expressed in the Serret-Frenet frame. It follows that

$$\dot{\boldsymbol{q}} = \mathbf{R}(\chi_f) \begin{bmatrix} \dot{s} \\ 0 \end{bmatrix}. \tag{69}$$

Subsequently, by substituting Eqs. (68) and (69) in Eq. (65), we obtain,

$$\dot{\boldsymbol{e}} = -\mathbf{S}(\dot{\chi}_f)\boldsymbol{e} + \mathbf{R}(\chi - \chi_f) \begin{bmatrix} V \\ 0 \end{bmatrix} - \begin{bmatrix} \dot{s} \\ 0 \end{bmatrix}, \tag{70}$$

where $\mathbf{R}^{\mathsf{T}}(\chi_f)\mathbf{R}(\chi) = \mathbf{R}(\chi - \chi_f)$.

Let now the error course angle $\widetilde{\chi}$ be defined by

$$\widetilde{\chi} \triangleq \chi - \chi_f. \tag{71}$$

Hence, we obtain the time derivative of $\widetilde{\chi}$,

$$\dot{\widetilde{\chi}} = \dot{\chi} - \dot{\chi}_f = \omega - \dot{\chi}_f, \tag{72}$$

where

$$\dot{\chi}_f \triangleq \frac{\mathrm{d}\chi_f}{\mathrm{d}t} = \frac{\mathrm{d}\chi_f}{\mathrm{d}s}\frac{\mathrm{d}s}{\mathrm{d}t} = \kappa(s)\dot{s}, \tag{73}$$

and $\kappa(s)$ is the curvature of the path at $\boldsymbol{q}(s)$.

The error kinematic model of a fixed-wing UAV for the path following problem with respect to the Serret-Frenet frame is summarized as follows,

$$\dot{e}_s = V\cos\widetilde{\chi} - (1 - \kappa(s)e_d)\dot{s}, \tag{74a}$$

$$\dot{e}_d = V\sin\widetilde{\chi} - \kappa(s)e_s\dot{s}, \tag{74b}$$

$$\dot{\widetilde{\chi}} = \omega - \kappa(s)\dot{s}. \tag{74c}$$

From Eqs. (74), the path following problem reduces into a problem of driving the errors to zero as the UAV approaches the given path. In Ref. [96], the point $\boldsymbol{q}$ is found by projecting $\boldsymbol{p}$ on the path, assuming the projection is well defined. Hence, the control law derived in [96] requires a stringent restriction on the initial position of $\boldsymbol{q}$ in order to avoid singularities. In contrast, Lapierre and Soetanto[75] proposed to employ a moving Serret-Frenet frame along the path, which effectively provides an extra control parameter $\dot{s}$ allowing $\boldsymbol{q}(s)$ to evolve along the error states. This control parameter then mitigates the stringent restriction on the initial condition arose in [96].

4.3 Path Following Controller Design

In this section we present a nonlinear path following control logic, which steers the UAV to the reference path with an inaccurately known time constant λ_ϕ. Beginning with the derivation of a kinematic control law for the heading rate command, we employ backstepping to derive a roll control command for a fixed-wing UAV, which, in turn, induces an equivalent control effort by the kinematic control law. In addition, we apply a parameter adaptation technique to deal with the inaccurately known time constant.

4.3.1 Kinematic controller design

Following a similar approach as in Ref. [75], we first derive a kinematic control law for the heading rate. We introduce a bounded differentiable function with respect to the cross-track error e_d, as follows

$$\delta(e_d) = -\chi^\infty \frac{e^{2ke_d} - 1}{e^{2ke_d} + 1} \tag{75}$$

where, $k > 0$ and $\chi^\infty \in (0, \pi/2)$[96]. This function is so-called the *approach angle*, since it provides the desired relative course transition of the UAV to the path as a function of the cross-track error e_d. When the cross-track error e_d is zero, the approach angle vanishes, thus imposing the condition that the course angle of the UAV must be tangential to the path. The positive constant k sets the effectiveness of the transient maneuver during approach. The approach angle provides a behavior similar to that of a 'good helmsman'[120] since it satisfies the following condition:

$$e_d \sin\big(\delta(e_d)\big) \leq 0, \qquad \text{for all } e_d. \tag{76}$$

This condition steers the UAV to the path along the correct direction (turn left when the UAV is on the right side of the path, and turn right in the opposite situation).

The kinematic controller for path following using the error kinematic model in Eq. (74) is given below.

Proposition 1 *Consider the kinematic error model of the UAV described in Eq. (74) and the approach angle $\delta(e_d)$ defined as in Eq. (75). Assume that the speed of the UAV is non-negative and suppose that the reference path is parameterized by the arc-length s, and that at each s the variables κ, e_s, e_d, and χ_f are well defined. Then, the following kinematic control law,*

$$\begin{aligned}\omega &= -k_\omega\big(\widetilde{\chi}-\delta(e_d)\big)+\kappa(s)\dot{s}+\delta'(e_d)\big(V\sin\widetilde{\chi}-\kappa(s)e_s\dot{s}\big)\\ &\quad -\frac{e_dV}{\gamma}\left(\frac{\sin\widetilde{\chi}-\sin\big(\delta(e_d)\big)}{\widetilde{\chi}-\delta(e_d)}\right), \qquad (77a)\\ \dot{s} &= k_se_s+V\cos\widetilde{\chi}, \qquad (77b)\end{aligned}$$

where k_s, k_ω and γ are positive constants, asymptotically drives e_s, e_d, and $\widetilde{\chi}$ towards zero.

Proof Let V_1 be a positive definite and radially unbounded Lyapunov function[96],

$$V_1=\frac{1}{2\gamma}\big(e_s^2+e_d^2\big)+\frac{1}{2}\big(\widetilde{\chi}-\delta(e_d)\big)^2. \qquad (78)$$

In the Lyapunov candidate function adopted, the first term captures the distance between the UAV and the path, and the second term intends to steer the error course angle $\widetilde{\chi}$ to the approach angle $\delta(e_d)$, which forces the UAV to follow a desired transition profile.

Differentiating V_1 with respect to time, we get

$$\begin{aligned}\dot{V}_1 &= \frac{1}{\gamma}\big(e_s\dot{e}_s+e_d\dot{e}_d\big)+\big(\widetilde{\chi}-\delta(e_d)\big)\big(\dot{\widetilde{\chi}}-\dot{\delta}(e_d)\big)\\ &= \frac{1}{\gamma}\Big\{e_s\big(V\cos\widetilde{\chi}-(1-\kappa(s)e_d)\dot{s}\big)+e_d\big(V\sin\widetilde{\chi}-\kappa(s)e_s\dot{s}\big)\Big\}\\ &\quad+\big(\widetilde{\chi}-\delta(e_d)\big)\Big\{\omega-\kappa(s)\dot{s}-\delta'(e_d)\big(V\sin\widetilde{\chi}-\kappa(s)e_s\dot{s}\big)\Big\}\\ &= \frac{1}{\gamma}e_s(V\cos\widetilde{\chi}-\dot{s})+\frac{1}{\gamma}e_dV\sin\big(\delta(e_d)\big)\\ &\quad+\big(\widetilde{\chi}-\delta(e_d)\big)\Big\{\omega-\kappa(s)\dot{s}-\delta'(e_d)\big(V\sin\widetilde{\chi}-\kappa(s)e_s\dot{s}\big)+\frac{e_dV}{\gamma}\frac{\sin\widetilde{\chi}-\sin\big(\delta(e_d)\big)}{\widetilde{\chi}-\delta(e_d)}\Big\}.\end{aligned} \qquad (79)$$

Using the control law for ω and $\dot{s}$ from Eqs. (77) yields,

$$\dot{V}_1 = -\frac{k_s}{\gamma} e_s^2 + \frac{e_d V}{\gamma} \sin\big(\delta(e_d)\big) - k_\omega \big(\widetilde{\chi} - \delta(e_d)\big)^2 \leq 0. \tag{80}$$

Note that V_1 is radially unbounded, hence the set $\Omega_c = \{V_1(e_s, e_d, \widetilde{\chi}) \leq c\}$ is a compact, positively invariant set. Let E_1 be the set of all points in Ω_c where $\dot{V}_1 = 0$. The set E_1 is given by $E_1 = \{(e_s, e_d, \widetilde{\chi}) \in \Omega_c | e_s = e_d = 0 \text{ and } \widetilde{\chi} = \delta\}$. Noticing that $\delta(\cdot)$ is a function of the cross-track error e_d, one can easily verify that any point starting from the set E_1 will remain in the set, or the set E_1 is an invariant set. Hence, by the LaSalle's invariance principle, we conclude that every trajectory starting in Ω_c approaches E_1 as $t \to \infty$. Therefore,

$$\lim_{t\to\infty} e_s = 0, \;\; \lim_{t\to\infty} e_d = 0, \text{ and } \lim_{t\to\infty} \widetilde{\chi} = \delta(e_d) \to 0, \tag{81}$$

since $e_d \to 0$ and $\delta(e_d) \to 0$. ∎

Remarks. The objective of the closed loop system is to steer the error states towards zero, so that the UAV follows the virtual target moving on the path. It should be noted that the approach angle plays an important role, in steering the UAV to the reference path by constructing a vector field with respect to a reference point on the path to guide the UAV to the path, as shown in Fig. 36. As mentioned earlier, the moving reference frame in conjunction with the extra control command $\dot{s}$ helps the errors converge to zero. In particular, the term $k_s e_s$ in Eq. (77b) ensures the convergence of e_s to zero, providing the momentum induced by the moving virtual target. Figure 36 shows the vector fields with respect to the corresponding reference points on the path, which takes into account the relative momentum induced by the virtual target (drawn by dashed arrows). During implementation, the new location of the virtual target on the path is propagated using the expression in Eq. (77b) via a numerical integration scheme, such as the forward Euler method. Hence, if the reference path is well defined in terms of the arc-length parameter s, then the

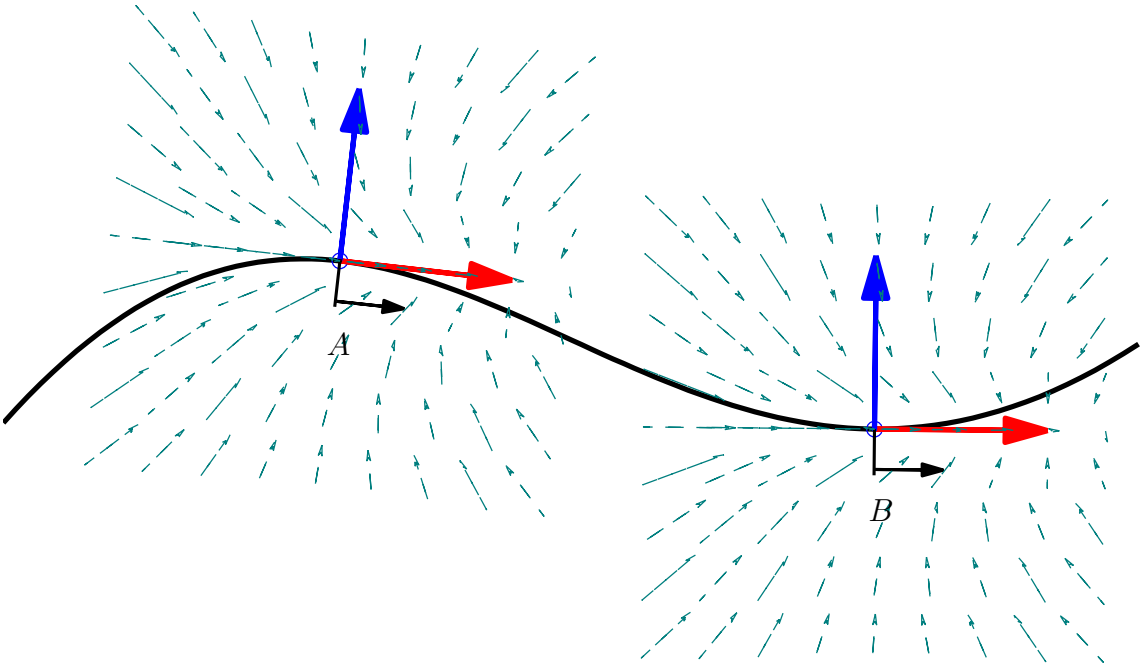


Figure 36: Local vector fields generated with respect to two distinct points on the path.

corresponding variables e_s, e_d, χ_f, and κ as also well defined, and thus the proposed path following algorithm is well-posed.

4.3.2 Roll angle command via backstepping

In this section, we apply backstepping to compute the roll control command from the heading rate command of the kinematic controller of the previous section.

For a fixed-wing UAV flying at a constant altitude, a roll angle command is often used for heading control. A simple way to compute the roll angle command directly from the heading rate command ω_d is using the kinematic relationship in Eq. (61),

$$\phi_d = \tan^{-1}\left(\frac{V\omega_d}{g}\right). \tag{82}$$

Note that the actual response of the roll angle differs from the desired value of Eq. (82), but it can be approximated by a first-order system described in Eq. (62). Taking into account the approximated model of Eq. (62), one can design a steering logic for the roll angle command ϕ_c in order to achieve $\omega \to \omega_d$ via $\phi \to \phi_d$. Following the standard

backstepping technique [134], we introduce an auxiliary control input ν for the roll angle by augmenting the error model in Eq. (74) with $\dot{\phi} = \nu$. Thus, the system is given as follows,

$$\dot{e}_s = V\cos\widetilde{\chi} - (1-\kappa(s)e_d)\dot{s}, \tag{83a}$$

$$\dot{e}_d = V\sin\widetilde{\chi} - \kappa(s)e_s\dot{s}, \tag{83b}$$

$$\dot{\widetilde{\chi}} = \frac{g}{V}\tan\phi - \kappa(s)\dot{s}, \tag{83c}$$

$$\dot{\phi} = \nu, \tag{83d}$$

where the auxiliary control input ν is associated with the roll command by

$$\nu = \frac{1}{\lambda_\phi}\big(\phi_c - \phi\big). \tag{84}$$

Suppose that the desired heading rate ω_d is obtained from Eq. (77a) and let $\omega_e \triangleq \omega - \omega_d = (g/V)\tan\phi - \omega_d$ be an error state for the heading rate, whose time derivative is given as follows

$$\dot{\omega}_e = \frac{g}{V}\dot{\phi}\sec^2\phi - \dot{\omega}_d. \tag{85}$$

Proposition 2 *Consider the kinematic error model of the UAV described in Eq. (83) and the approach angle $\delta(e_d)$ defined in Eq. (75). Assume that the speed of the UAV is non-negative and suppose that the reference path is parameterized by the arc-length s, and at each s the variables κ, e_s, e_d, and χ_f are well defined. Then, the control input,*

$$\nu = \frac{V}{g\sec^2\phi}\bigg(-k_e\omega_e - \widetilde{\chi} + \delta(e_d) + \dot{\omega}_d\bigg), \tag{86}$$

where k_e is a positive constant and ω_d is given in Eq. (77a), asymptotically drives e_s, e_d, and $\widetilde{\chi}$ towards zero, while $\omega \to \omega_d$.

Proof Let V_2 be an augmented Lyapunov candidate function given by,

$$V_2 = V_1 + \frac{1}{2}\omega_e^2. \tag{87}$$

where V_1 is given in Eq. (78). Differentiating with respect to time, one obtains

$$\begin{aligned}\dot{V}_2 &= \dot{V}_1 + \omega_e\dot{\omega}_e \\ &= \frac{1}{\gamma}e_s(V\cos\widetilde{\chi} - \dot{s}) + \frac{1}{\gamma}e_d V\sin\big(\delta(e_d)\big) \\ &\quad + \big(\widetilde{\chi} - \delta(e_d)\big)\bigg\{\frac{g}{V}\tan\phi + \omega_d - \omega_d - \kappa(s)\dot{s} - \delta'(e_d)\big(V\sin\widetilde{\chi} - \kappa(s)e_s\dot{s}\big) \\ &\quad + \frac{e_d V}{\gamma}\frac{\sin\widetilde{\chi} - \sin\big(\delta(e_d)\big)}{\widetilde{\chi} - \delta(e_d)}\bigg\} + \omega_e\bigg(\frac{g}{V}\dot{\phi}\sec^2\phi - \dot{\omega}_d\bigg).\end{aligned} \tag{88}$$

By the definition of ω_e, and substituting the desired ω_d from Eq. (77a) and $\dot{s}$ from Eq. (77b), the term inside of the bracket collapses to

$$\dot{V}_2 = -\frac{k_s}{\gamma}e_s^2 + \frac{e_d V}{\gamma}\sin\big(\delta(e_d)\big) - k_\omega\big(\widetilde{\chi} - \delta(e_d)\big)^2 + \omega_e\bigg(\frac{g}{V}\dot{\phi}\sec^2\phi + \widetilde{\chi} - \delta(e_d) - \dot{\omega}_d\bigg). \tag{89}$$

Choosing $\dot{\phi}$ through the auxiliary control input in Eq. (86), that is,

$$\dot{\phi} = \nu = \frac{V}{g\sec^2\phi}\bigg(-k_e\omega_e - \widetilde{\chi} + \delta(e_d) + \dot{\omega}_d\bigg), \tag{90}$$

results in,

$$\dot{V}_2 = -\frac{k_s}{\gamma}e_s^2 + \frac{e_d V}{\gamma}\sin\big(\delta(e_d)\big) - k_\omega\big(\widetilde{\chi} - \delta(e_d)\big)^2 - k_e\omega_e^2 \leq 0. \tag{91}$$

The last inequality implies, in particular, that $e_s, e_d, \widetilde{\chi}$ and ω_e are bounded. Furthermore, $\delta(e_d)$ is also bounded from Eq. (75) since e_d is bounded. It is also easy to show, using Eq. (77a), that ω_d is bounded. To this end, notice that all signals in the rhs of Eq. (77a) except the last are bounded. Moreover, since the last term is arranged by

$$\frac{\sin\widetilde{\chi} - \sin\big(\delta(e_d)\big)}{\widetilde{\chi} - \big(\delta(e_d)\big)} = \operatorname{sinc}\bigg(\frac{1}{2}\big(\widetilde{\chi} - \delta(e_d)\big)\bigg)\cos\bigg(\frac{1}{2}\big(\widetilde{\chi} + \delta(e_d)\big)\bigg) \tag{92}$$

and the sinc function is bounded, the last term in the rhs of Eq. (77a) is also bounded. Since ω_d and ω_e are bounded, it follows that $|\phi| < \pi/2$ and the control Eq. (86) is well defined.

To complete the proof, note that V_2 is radially unbounded, hence the set $\Omega_d = \{V_2(e_s, e_d, \widetilde{\chi}, \omega_e) \leq c\}$ is a compact, positively invariant set. Let E_2 be the set of

all points in Ω_d where $\dot{V}_2 = 0$. The set E_2 is given by $E_2 = \{e_s = e_d = 0,\ \widetilde{\chi} = \delta, \text{and } (g/V)\tan\phi = \omega_d\}$. Noticing that $\delta(\cdot)$ is a function of the cross-track error e_d, one can easily verify that any point starting from the set E_2 will remain in the set, i.e., E_2 is an invariant set. By the LaSalle's invariance principle, we have that every trajectory starting inside the set Ω_d approaches E_2 as $t \to \infty$. Therefore, $\lim_{t\to\infty} e_s = 0$, $\lim_{t\to\infty} e_d = 0$, and $\lim_{t\to\infty} \widetilde{\chi} = 0$ since $\delta(e_d) \to 0$. We also have $(g/V)\tan\phi \to \omega_d$ as $t \to \infty$. ∎

4.3.3 Parameter adaptation

Note that the actual roll angle command is computed from Eq. (62) and (86),

$$\phi_c = \lambda_\phi \nu + \phi, \tag{93}$$

where, λ_ϕ is the time constant of the system.

In practice, λ_ϕ is determined by the characteristics of the UAV airframe and PID gains, thus it is not known accurately. The actual roll command $\widehat{\phi}_c$ is then computed by

$$\widehat{\phi}_c = \widehat{\lambda}_\phi \nu + \phi, \tag{94}$$

where $\widehat{\lambda}_\phi$ is an estimate of λ_ϕ.

In order to compensate for any uncertainty in λ_ϕ, we apply a parameter adaptation technique. To this end, let V_3 be a candidate Lyapunov function defined by

$$V_3 \triangleq V_2 + \frac{1}{2}\frac{1}{k_a \lambda_\phi}\widetilde{\lambda}_\phi^2, \tag{95}$$

where V_2 is given in Eq. (87) and $\widetilde{\lambda}_\phi \triangleq \lambda_\phi - \widehat{\lambda}_\phi$ is the parameter estimate error. Differentiating with respect to time, we get

$$\begin{aligned}
\dot{V}_3 &= \dot{V}_2 + \frac{1}{k_a \lambda_\phi}\widetilde{\lambda}_\phi \dot{\widetilde{\lambda}}_\phi \\
&= -\frac{k_s}{\gamma}e_s^2 + \frac{e_d V}{\gamma}\sin\big(\delta(e_d)\big) - k_\omega\big(\widetilde{\chi} - \delta(e_d)\big)^2 \\
&\quad + \omega_e\left(\frac{g}{V}\dot{\phi}\sec^2\phi + \widetilde{\chi} - \delta(e_d) - \dot{\omega}_d\right) + \frac{1}{k_a \lambda_\phi}\widetilde{\lambda}_\phi \dot{\widetilde{\lambda}}_\phi.
\end{aligned} \tag{96}$$

The actual value of $\dot{\phi}$ is calculated from Eq. (62) in conjunction with the actual roll command in Eq. (94) and ν given in Eq. (86). Subsequently, substituting the actual $\dot{\phi}$ into Eq. (96), we get

$$\begin{aligned}\dot{V}_3 = -\frac{k_s}{\gamma}e_s^2 + \frac{e_d V}{\gamma}\sin\big(\delta(e_d)\big) - k_\omega\big(\widetilde{\chi} - \delta(e_d)\big)^2 - \frac{\widehat{\lambda}_\phi}{\lambda_\phi}k_e\omega_e^2 \\ + \omega_e\frac{\widetilde{\lambda}_\phi}{\lambda_\phi}\big(\widetilde{\chi} - \delta(e_d) - \dot{\omega}_d\big) + \frac{1}{k_a\lambda_\phi}\widetilde{\lambda}_\phi\dot{\widetilde{\lambda}}_\phi.\end{aligned} \tag{97}$$

Choose $\dot{\widetilde{\lambda}}_\phi$ as,

$$\dot{\widetilde{\lambda}}_\phi = -k_a\omega_e\big(\widetilde{\chi} - \delta(e_d) - \dot{\omega}_d\big), \tag{98}$$

where k_a is a positive constant. It follows that

$$\dot{V}_3 = -\frac{k_s}{\gamma}e_s^2 + \frac{e_d V}{\gamma}\sin\big(\delta(e_d)\big) - k_\omega\big(\widetilde{\chi} - \delta(e_d)\big)^2 - \frac{\widehat{\lambda}_\phi}{\lambda_\phi}k_e\omega_e^2 \leq 0. \tag{99}$$

Assuming λ_ϕ is constant, the parameter update law is readily obtained from Eq. (98) as follows,

$$\dot{\widehat{\lambda}}_\phi = k_a\omega_e\big(\widetilde{\chi} - \delta(e_d) - \dot{\omega}_d\big). \tag{100}$$

Proposition 3 *Let the control law in Eqs. (77) and (86). With the parameter update law given by Eq. (100) the actual roll command of Eq. (94) guarantees that the signals e_s, e_d, and $\widetilde{\chi}$ asymptotically tend to zero, while $(g/V)\tan\phi \to \omega_d$.*

Proof In order to prove the proposition notice that $\lim_{t\to\infty} V_3(t)$ exists since V_3 is bounded from below and is non-increasing. It therefore suffices to show that $\dot{V}_3$ is uniformly continuous, in which case we can invoke Barbalat's lemma to ensure that $\lim_{t\to\infty}\dot{V}_3(t) = 0$.

To this end, consider the expression of $\dot{V}_3$ in Eq. (99), from which it follows that e_d, e_s, $\widetilde{\chi}$, ω_e, $\widehat{\lambda}_\phi$, and $\widetilde{\lambda}_\phi$ are all bounded. As with the proof of Proposition 2, it follows from (77b) that $\dot{s}$ is bounded, and from (77a) that ω_d is bounded as well. The boundedness of ω_d and ω_e imply that $|\phi| < \pi/2$. Using (83) we conclude that $\dot{e}_s, \dot{e}_d, \dot{\widetilde{\chi}}$ are also bounded. Differentiating Eq. (77a) with respect to time, one can show that

all terms are bounded, while the last term is also bounded since the derivative of the sinc function is bounded (see also Eq. (92)). Hence, it can be shown that $\dot{\omega}_d$ is bounded. Furthermore, $\delta(e_d)$ and its time derivative $\dot{\delta}(e_d) = \delta'(e_d)\dot{e}_d$ are bounded. A straightforward calculation shows that

$$\dot{\omega}_e = \frac{\widehat{\lambda}_\phi}{\lambda_\phi}\big(- k_e\omega_e - \widetilde{\chi} + \delta(e_d)\big) - \frac{\widetilde{\lambda}_\phi}{\lambda_\phi}\dot{\omega}_d. \tag{101}$$

From the previous equation it follows that $\dot{\omega}_e$ is bounded. Furthermore, $\dot{\widehat{\lambda}}_\phi$ is also bounded from (100). Consider now the expression for $\ddot{V}_3$, given by

$$\begin{aligned}\ddot{V}_3 = &-\frac{k_s}{\gamma}e_s\dot{e}_s + \frac{V\dot{e}_d}{\gamma}\sin\big(\delta(e_d)\big) + \frac{Ve_d}{\gamma}\dot{\delta}(e_d)\cos\big(\delta(e_d)\big) \\ &- 2k_\omega\big(\widetilde{\chi} - \delta(e_d)\big)\big(\dot{\widetilde{\chi}} - \dot{\delta}(e_d)\big) - \frac{\widehat{\lambda}_\phi}{\lambda_\phi}k_e\omega_e\dot{\omega}_e,\end{aligned} \tag{102}$$

where, all expressions in the rhs of the equation have been shown to be bounded. It follows that $\ddot{V}_3$ is bounded and hence $\dot{V}_3$ is uniformly continuous. Applying Barbalat's lemma, it follows that $\dot{V}_3 \to 0$ as $t \to 0$. ∎

4.4 Simulation results

This section illustrates the performance of the derived path-following control law. The reference path is given by a quartic B-spline over a non-decreasing knot parameter u which is monotonic to the arc-length s. Hence, we can compute $\boldsymbol{q}(s)$, $\chi_f(s)$, and $\kappa(s)$ in terms of the knot parameter as follows,

$$\chi_f(s) = \tan^{-1}\frac{y'}{x'}\ , \quad \kappa(s) = \frac{x'y'' - x''y'}{(x'^2 + y'^2)^{3/2}}. \tag{103}$$

where $(\cdot)'$ and $(\cdot)''$ are the derivatives with respect to u. The knot parameter is propagated along with Eq. (77b) as follows,

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \frac{\mathrm{d}s}{\mathrm{d}t}\Big/\frac{\mathrm{d}s}{\mathrm{d}u} = \dot{s}/\sqrt{x'^2 + y'^2}. \tag{104}$$

The parameters used in the simulations are shown in Table 5. We present the results from two simulations. In the first case, we calculate the roll angle command from

Table 5: Simulation parameters.

V = 20 [m/s]	$k = 0.01$
$k_s = 0.4$	$k_\omega = 0.001$
$\lvert\phi_c^{\max}\rvert = \pi/6$	$k_e = 1.1$
$k_a = 0.7$	$\gamma = 4000$
$\lambda_\phi \approx 1.1$	

Eq. (93) using a time constant $\lambda_\phi = 0.5$ which is known inaccurate. Without parameter adaptation, as shown in Fig. 37, we observe a sluggish and low damped response of the actual trajectory. The error variables are shown in Fig. 38, and the command inputs are shown in Fig. 39.

In the second case, we calculate the roll angle command from Eq. (94) with the parameter update law of Eq. (100). Figures 40-43 show the results. The error states tend to zero asymptotically as shown in Fig. 41. As the command inputs are shown in Fig. 42, the roll angle command is limited within $\pm\pi/6$, hence the UAV is unable to exactly follow the path where the curvature exceeds the maximum curvature achievable by the UAV at the speed of 20 [m/sec]. Nevertheless, the path following control law forces the UAV converge to the path asymptotically after a short transient. Finally, Fig. 43 displays the time history of the estimate of λ_ϕ.

4.5 Summary

In this chapter a nonlinear path following control law has been developed for a small UAV using backstepping of the heading rate command. The kinematic control law realizes cooperative path following control such that the motion of a virtual target is controlled by an extra control input to help the convergence of the error variables. A roll command that gives rise to the desired heading rate has been derived by taking into account the inaccurate system time constant with parameter adaptation scheme. From a high-fidelity hardware-in-the-loop (HIL) simulation results, the presented algorithm is validated for the applicability to the actual UAV.

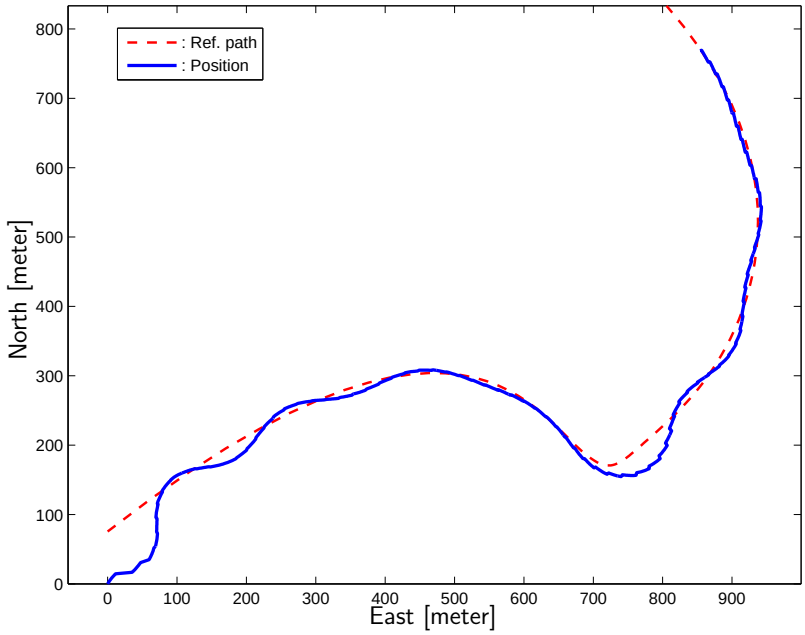
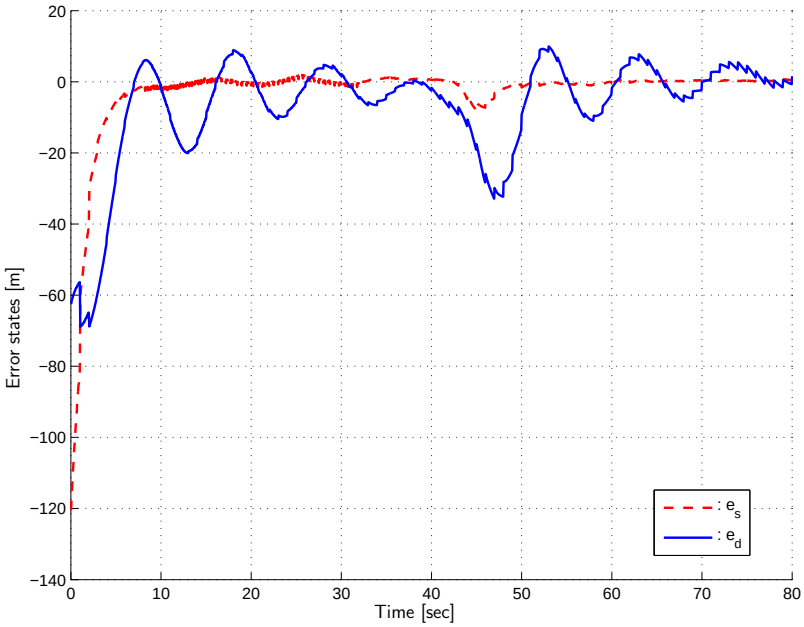
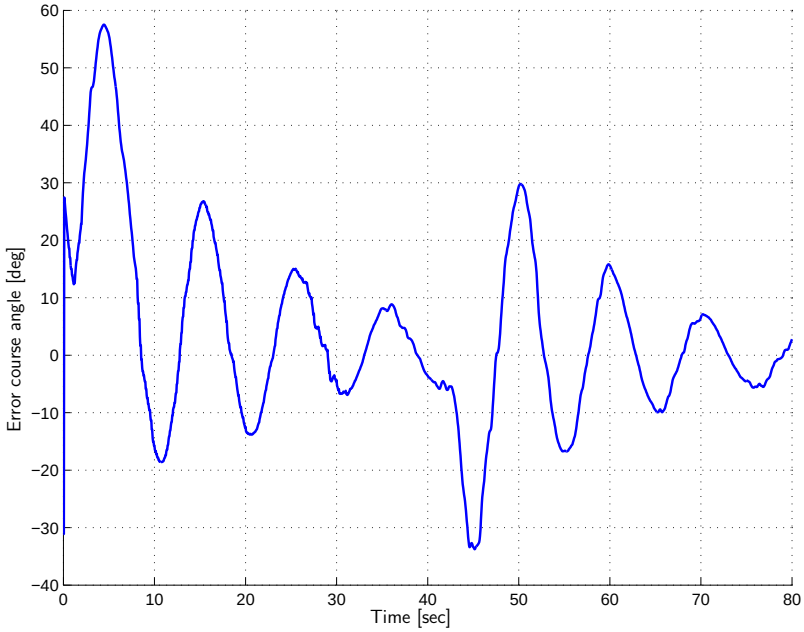


Figure 37: Reference path and actual trajectory of the UAV without parameter adaptation.

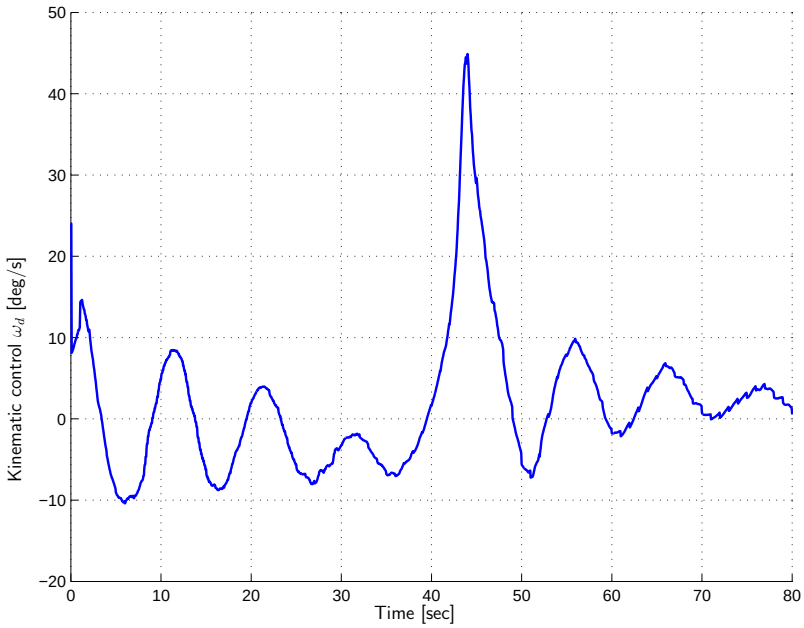


(a) Track errors

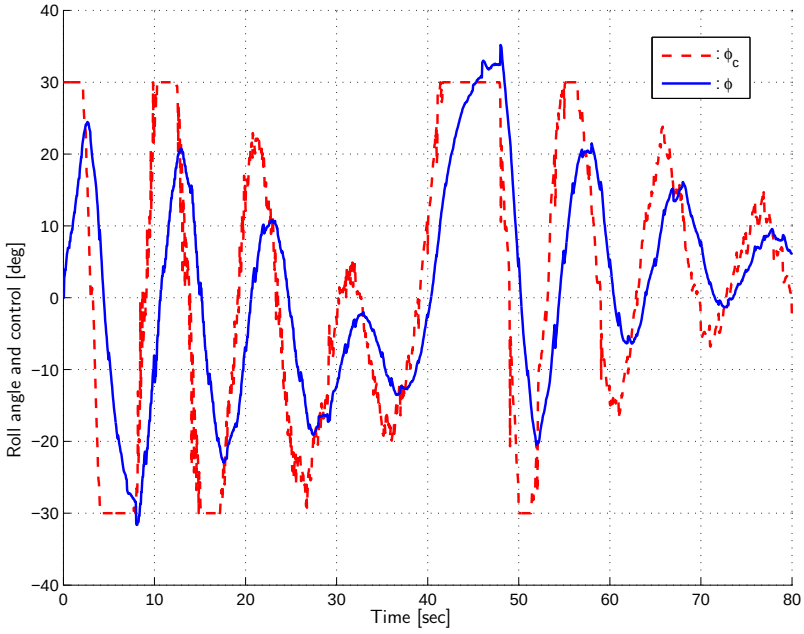


(b) Course angle error

Figure 38: Error states without parameter adaptation.



(a) Heading rate command



(b) ϕ v/s ϕ_c

Figure 39: Command inputs without parameter adaptation.

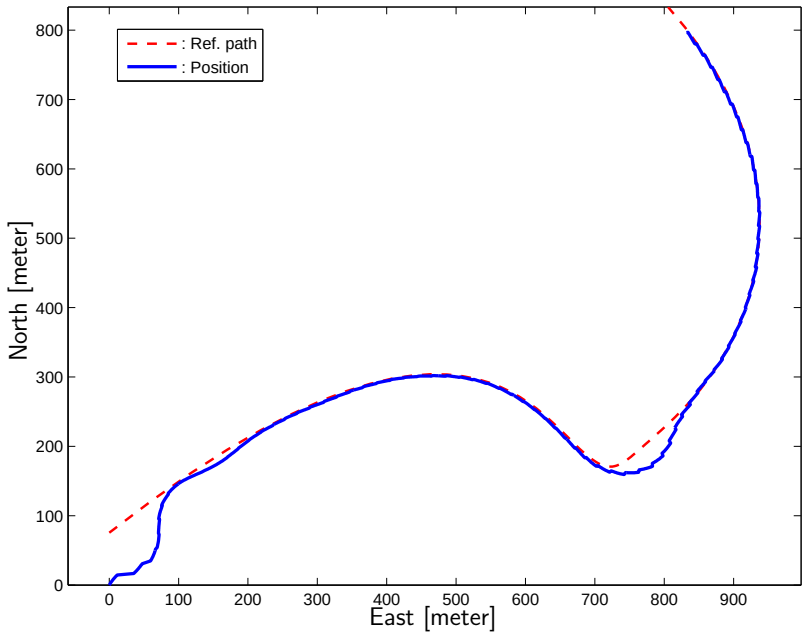
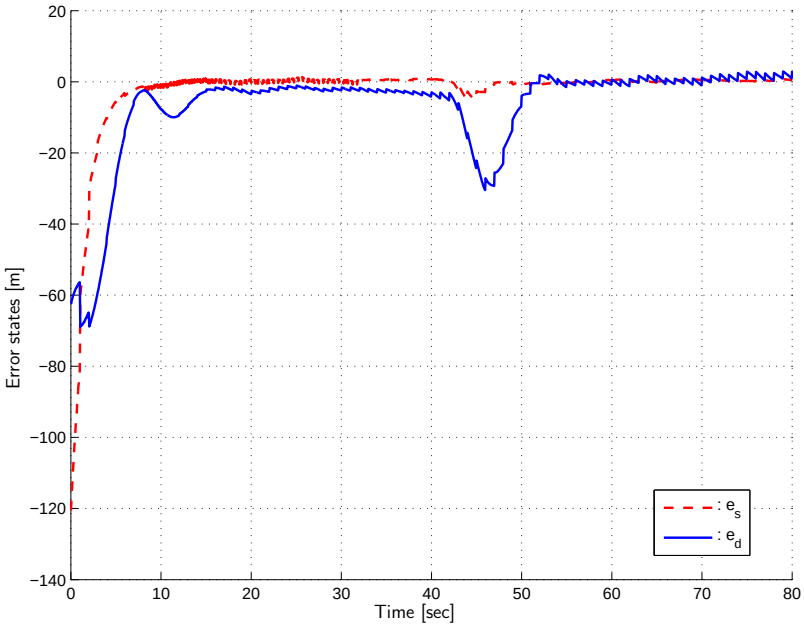
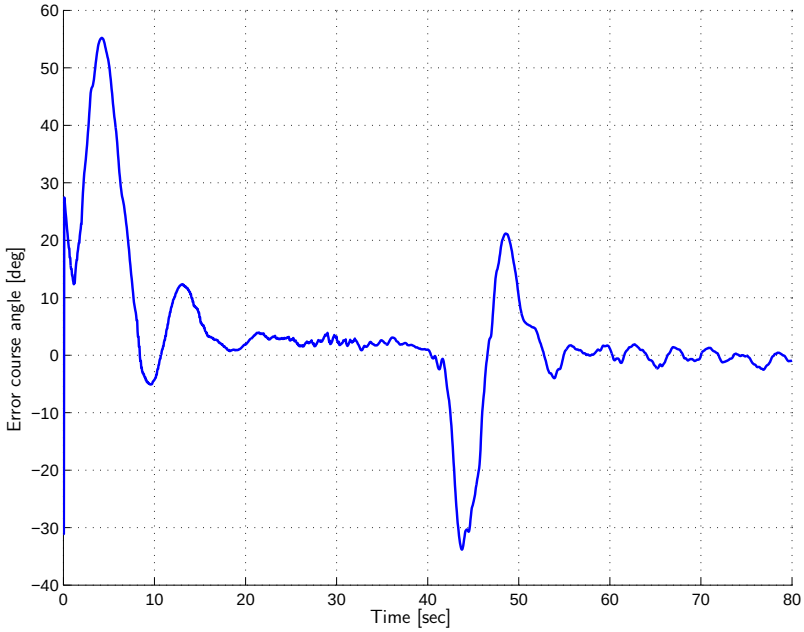


Figure 40: Reference path and actual trajectory of the UAV with parameter adaptation.

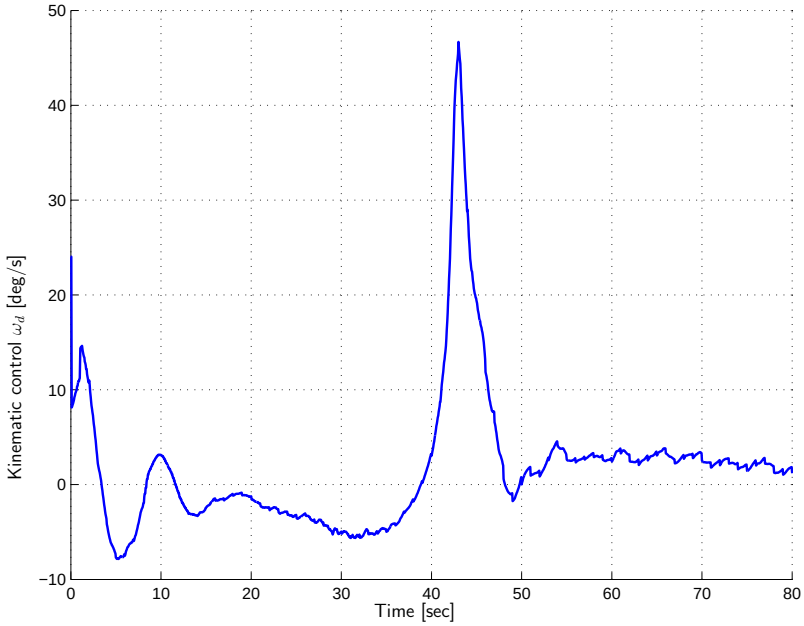


(a) Track errors

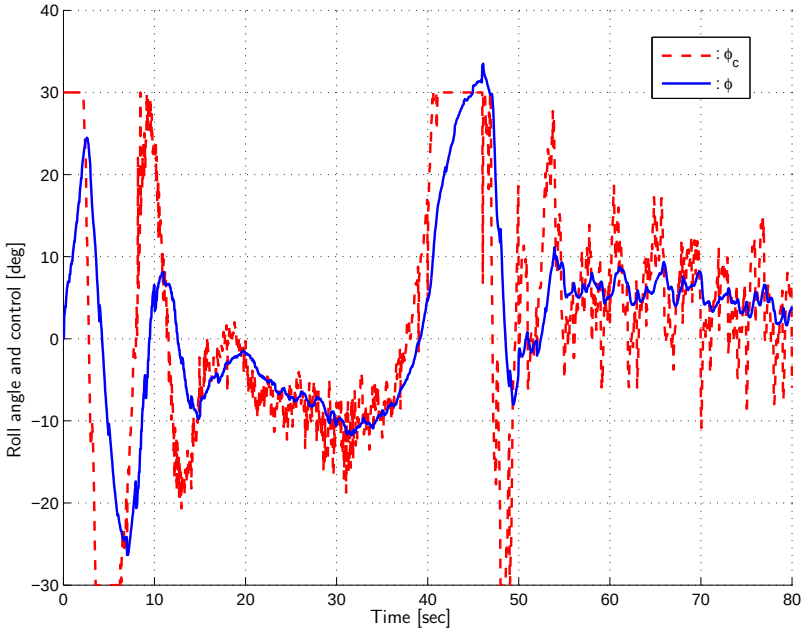


(b) Course angle error

Figure 41: Error states with parameter adaptation.



(a) Heading rate command



(b) ϕ v/s ϕ_c

Figure 42: Command inputs with parameter adaptation.

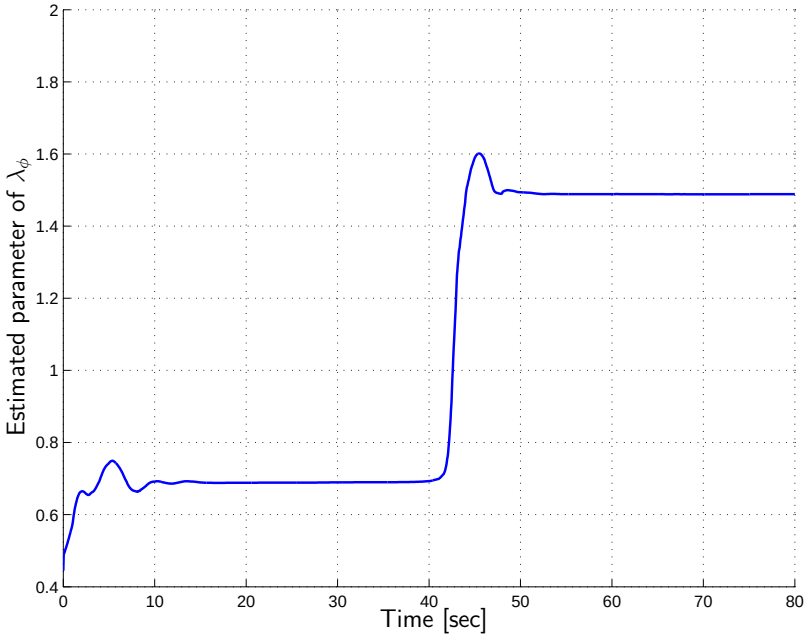


Figure 43: Parameter estimate of λ_ϕ.

CHAPTER V

REAL TIME IMPLEMENTATION OF THE HIERARCHICAL PATH CONTROL ALGORITHM USING HARDWARE-IN-THE-LOOP SIMULATION

In this chapter we present on-line, real-time hardware-in-the-loop simulation results of the hierarchical path control algorithm using a small micro-controller. The control hierarchy, as depicted in Fig. 44, consists of path planning (Chapter 2), path smoothing (Chapter 3), and path following control (Chapter 4).

At each stage of the control hierarchy, the required commands for the next stage are calculated, which finally drives the control surfaces of the fixed-wind UAV. The information regarding the world is initially given to the autopilot by a two dimensional elevation map. With the given initial position of the UAV, the user can choose any arbitrary goal position for the UAV to fly safely. Details of the implementation are discussed in the sequel.

5.1 Hardware description

A UAV research platform, as shown in Fig. 45, has been developed to support the UAV research. The development of the hardware and software was done completely in-house to have a full access to the entire system. The on-board autopilot is equipped with a micro-controller, sensors and actuators, along with the communication devices. The micro-controller adopted in this research was chosen to be a Rabbit RCM-3400 module which runs at 30 MHz clock speed equipped with 512 KB RAM for data and 512 KB ROM for program.

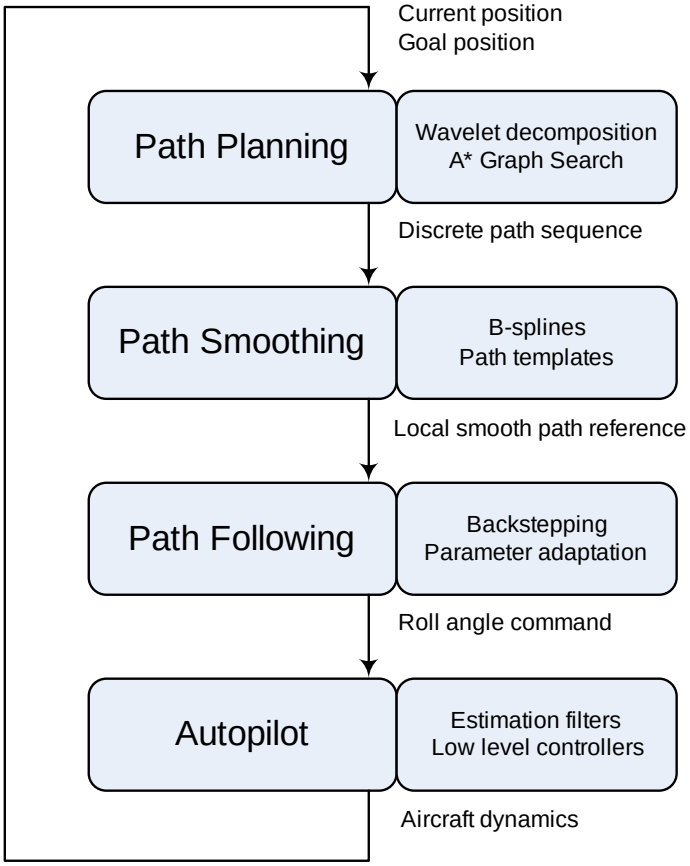


Figure 44: Block diagram for control hierarchy of the proposed path control algorithm.

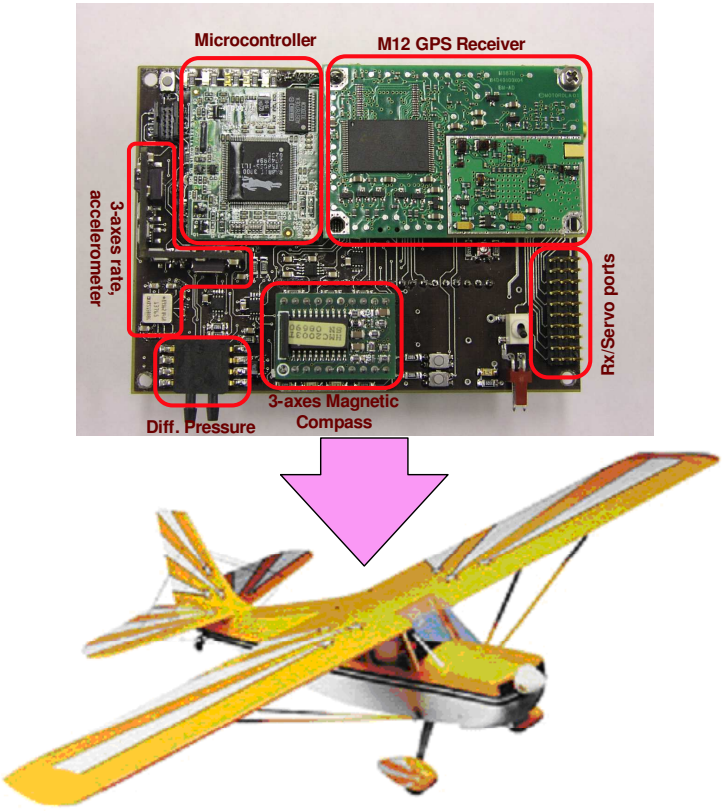


Figure 45: A small fixed-wing UAV equipped with an autopilot for hierarchical path planning control.

5.2 Hardware-in-the-loop simulation environment

A realistic hardware-in-the-loop simulation (HILS) environment has also been developed to validate the UAV autopilot hardware and software development utilizing Matlab® and Simulink®. A full 6-DOF nonlinear aircraft model is used in conjunction with a linear approximation of the aerodynamic forces and moments, along with Earth gravitational (WGS-84) and magnetic field models. Detailed models for the sensors and actuators have also been incorporated. Four independent computer systems are used in the hardware-in-the-loop simulation (HILS) as illustrated in Fig. 46. A 6-DOF simulator, the flight visualization computer, the autopilot micro-controller, and the ground station computer console are involved in the simulation. Further details about the UAV platform, autopilot and HILS set-up are described in the

Appendices.

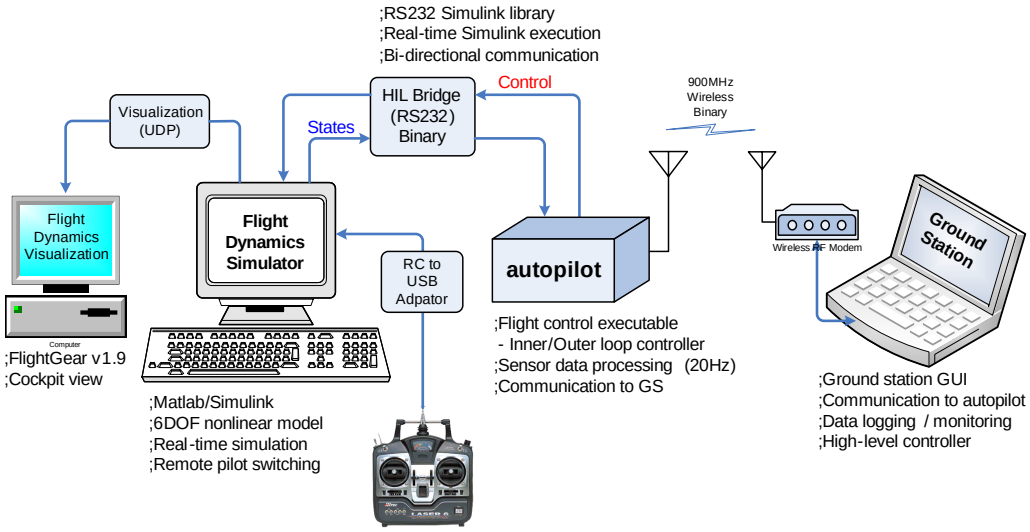
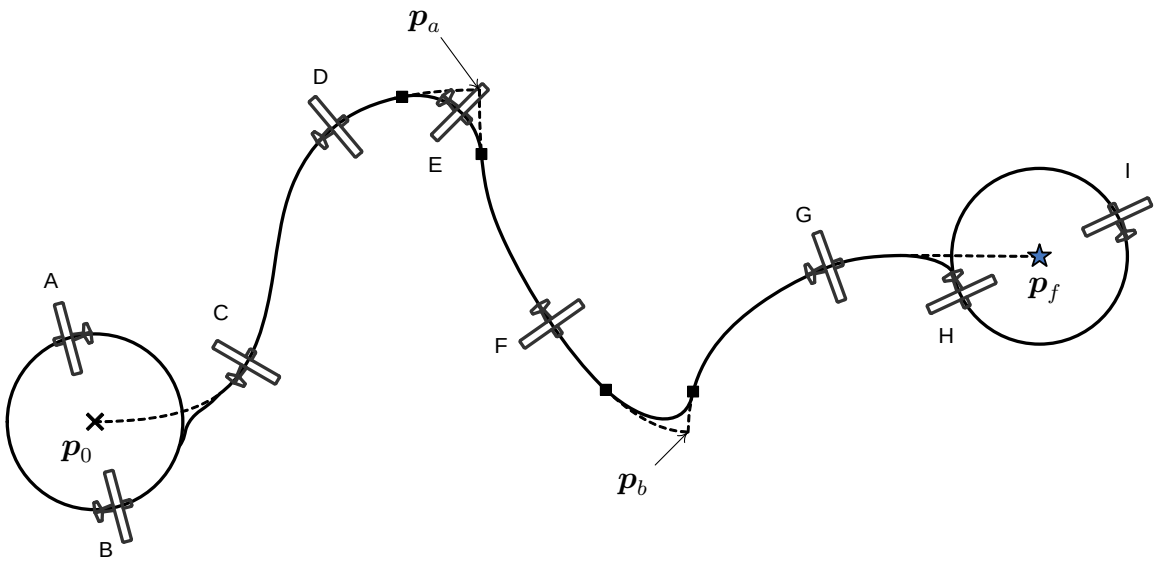


Figure 46: High fidelity hardware-in-the-loop simulation (HILS) environment that enables rapid testing of the proposed path planning algorithm.

5.3 Simulation scenario

The environment $\mathcal{W}$ is the elevation map of a certain area in US state. The environment is assumed to be square of dimension 128×128 units, which corresponds to 9600×9600 meters in actual size. Taking into account the available memory of the micro-controller, we choose the fine resolution level as $J_{\max} = 6$ and the coarser resolution level as $J_{\min} = 3$. Cells at the fine resolution have dimensions 150×150 meters, which is slightly larger than the minimum turning radius of the fixed-wing UAV. The minimum turning radius is approximately calculated for the UAV flying at the constant speed of $V_T = 20$ [m/sec] with the bounded roll angle $|\phi| \leq 30°$, resulting in $R_{\min} \approx 70$ [meter].

The objective of the UAV is to follow a path from the initial position to the final position while circumventing the obstacles over a certain elevation threshold. Figure 47 illustrates the detail on-line implementation of the proposed path control algorithm. Initially, the UAV is loitering around the initial position $\boldsymbol{p}_0$ until when



Step	Task description
A	Initially, the UAV is loitering around the initial position with the circle radius R_l
B	Calculate the the first path segment from $\boldsymbol{p}_0$ to $\boldsymbol{p}_a$
C	Break away from the loiter circle, start to follow the first path segment
D	Calculate the second path segment from $\boldsymbol{p}_a$ to $\boldsymbol{p}_b$, and a transient path connecting two paths
E	UAV is on the transient path
F	Calculate the third path segment, and a transient path
G	UAV is approaching the goal position, no path calculated
H	The goal is reached, end of the path control, get on the loitering circle
I	UAV is loitering around the goal position $\boldsymbol{p}_f$

Figure 47: Illustration of the on-line implementation of the proposed hierarchical path control algorithm.

a local path segment from $\boldsymbol{p}_0$ to $\boldsymbol{p}_a$ is computed (Step A,B). Subsequently, the path following controller is engaged to follow the path (Step C,D). At step D, the UAV replans to compute a new path from the intermediate location $\boldsymbol{p}_a$ to the goal, resulting in the second local path segments from $\boldsymbol{p}_a$ to $\boldsymbol{p}_b$. The first and second path segments are stitched by a transient B-spline path assuring the continuity condition at each intersection points (marked by black squares). This process iterates until the final position $\boldsymbol{p}_f$ is reached (Step H), when the UAV engages to loiter around the goal location.

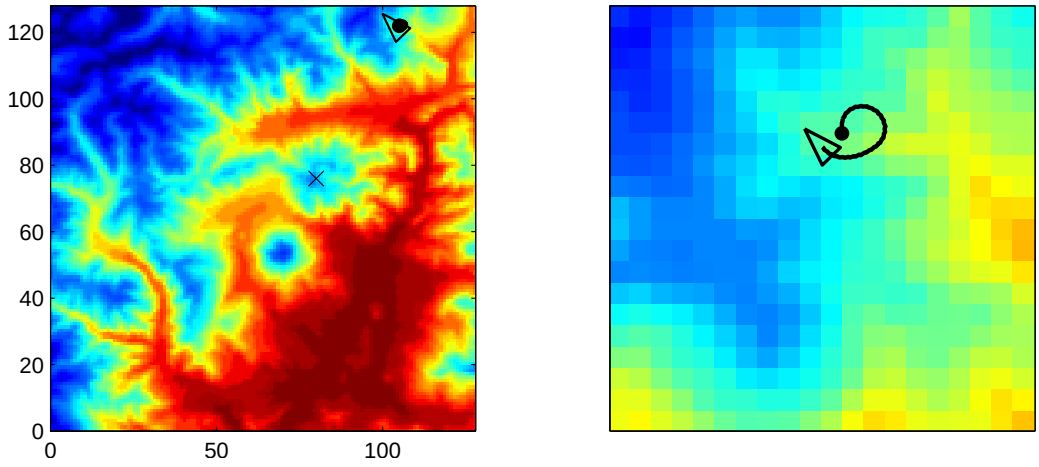
5.4 Simulation results

Figure 48 shows the simulation results of the hierarchical path control implementation. Specifically, figures on the right side show the close-up view of the simulation. The channels are drawn by polygonal lines, and the UAV smoothly follows the reference path avoiding the possible obstacles outside the channels. Consequently, the UAV reaches the final destination in a collision free manner, as seen in Fig. 48(o).

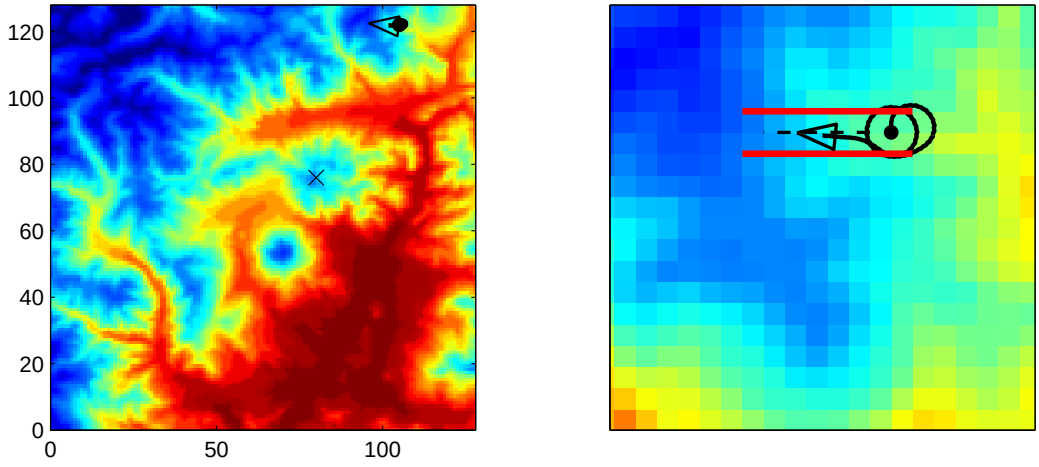
Figure 49 shows a 3D screen shot during the simulation. The ground track of the followed path is displayed showing that the UAV is avoiding the obstacles (for this case, it is the high elevation region) effectively.

5.5 Summary

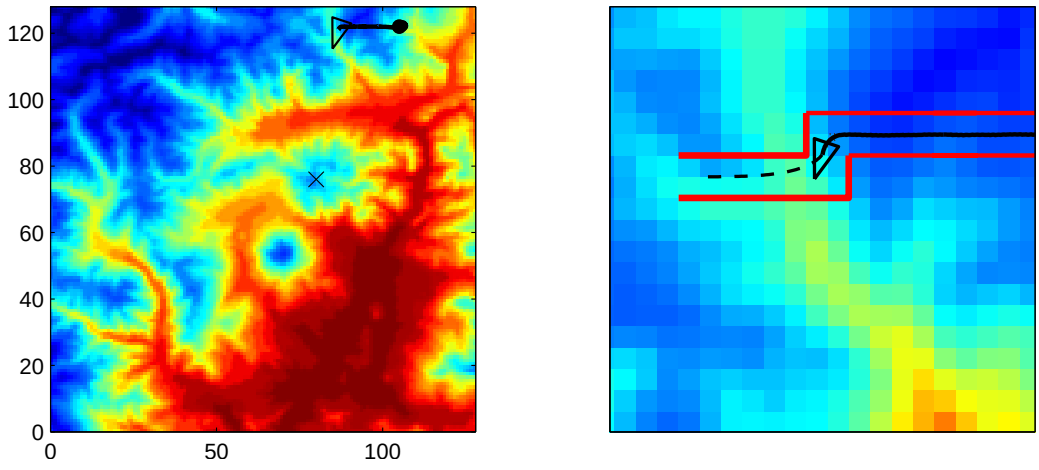
We implement the hierarchical path control of a small UAV on the actual hardware platform. Based on the high fidelity hardware-in-the-loop (HIL) simulation environment, the proposed hierarchical path control algorithm has been validated through the on-line, real-time implementation on a small micro-controller. By a seamless integration of the control algorithms for path planning, path smoothing, and path following, it has been demonstrated that the UAV equipped with a small autopilot having limited computational resources manages to accomplish the path control objective to reach the goal while avoiding obstacles with minimal human intervention.



(a) $t = 27.5$ [sec]

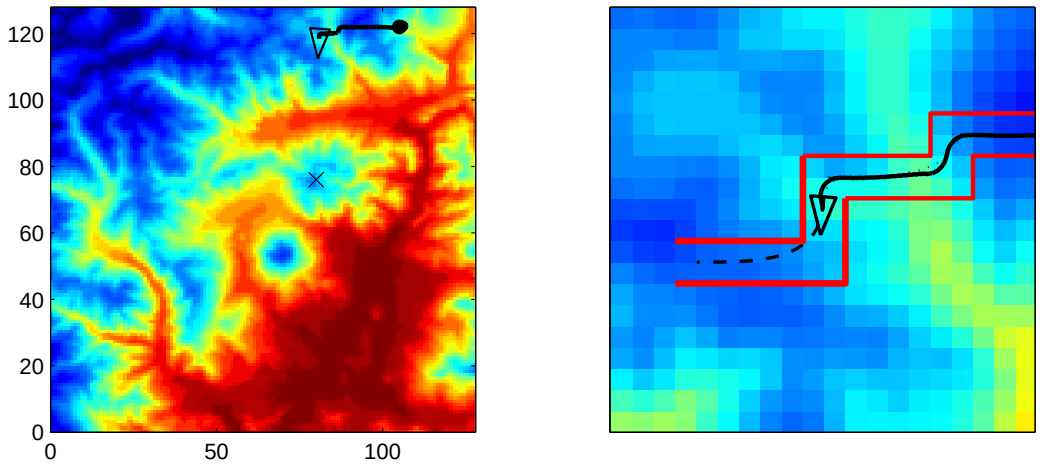


(b) $t = 64.5$ [sec]

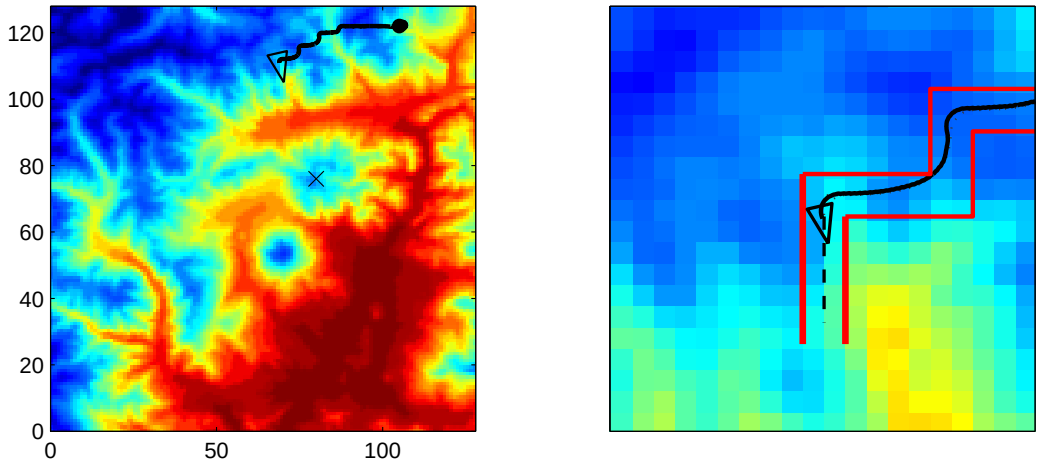


(c) $t = 126.5$ [sec]

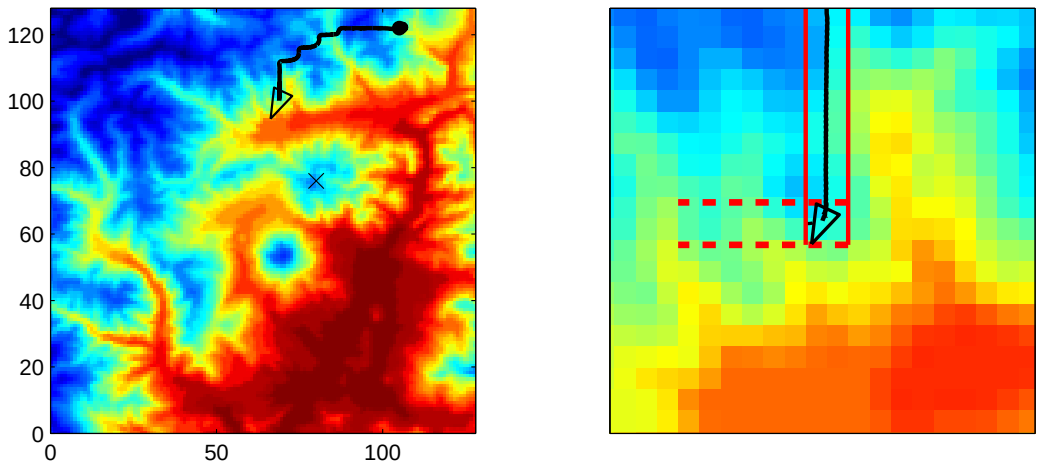
Figure 48: Simulation results of the hierarchical path control implementation. Figures on the right show the close-up view of the simulation. At each instant the channel is drawn by polygonal lines, where the smooth path segment from the path templates stays. The actual path followed by the UAV is drawn on top of the reference path.



(d) $t = 156.5$ [sec]

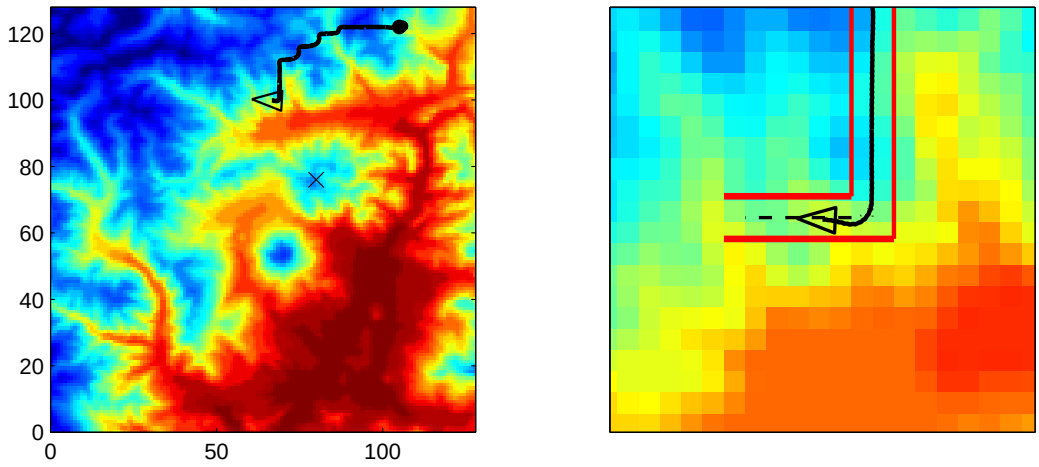


(e) $t = 222.0$ [sec]

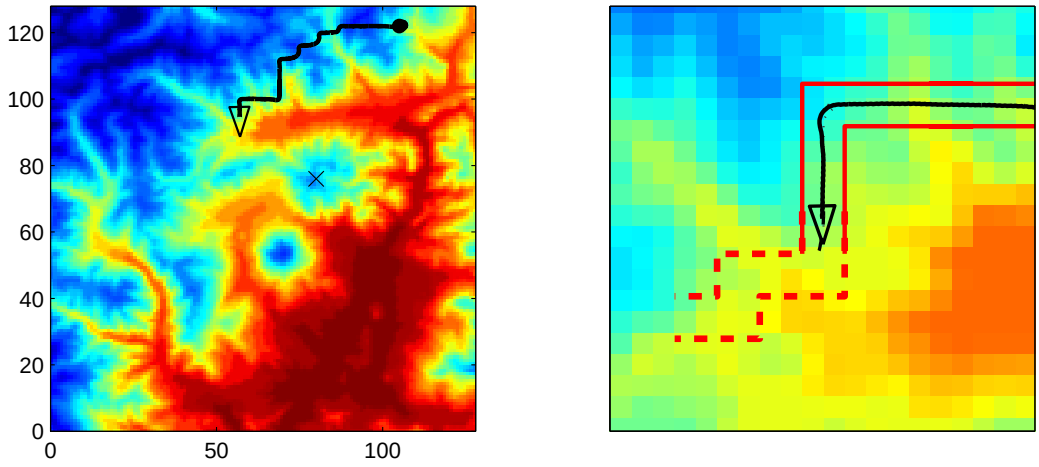


(f) $t = 265.0$ [sec]

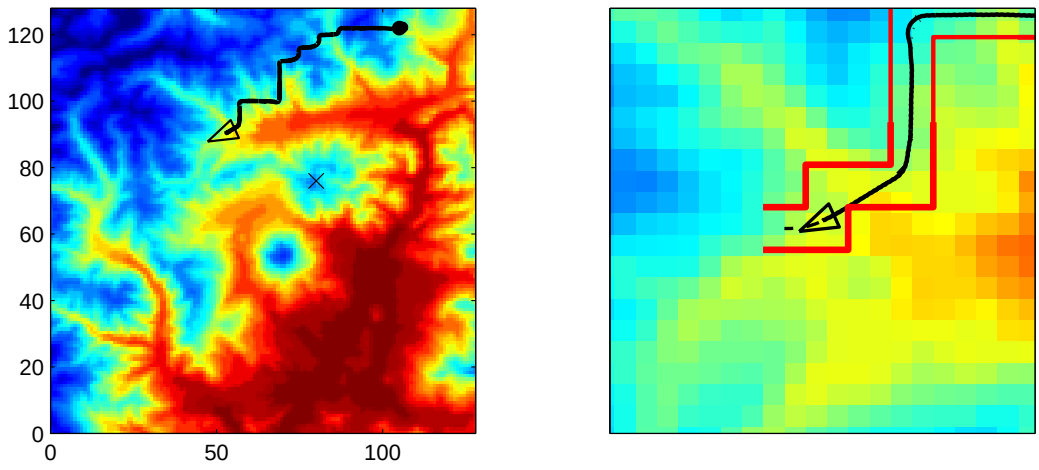
Figure 48: Simulation results of the hierarchical path control implementation. (cont'd)



(g) $t = 274.0$ [sec]

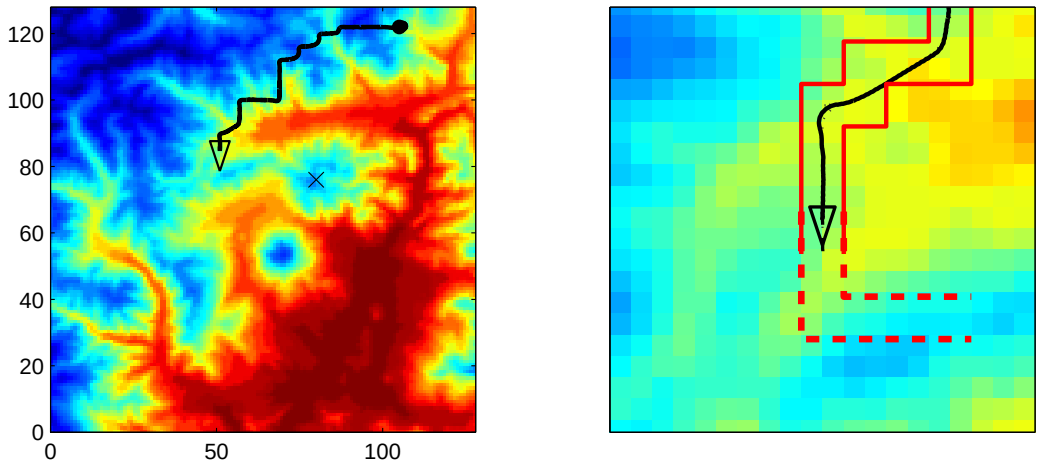


(h) $t = 333.0$ [sec]

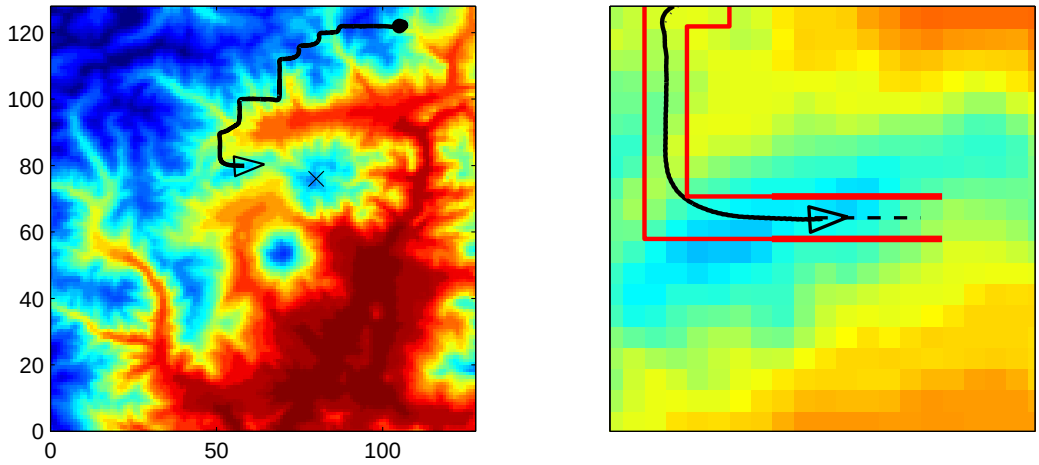


(i) $t = 358.5$ [sec]

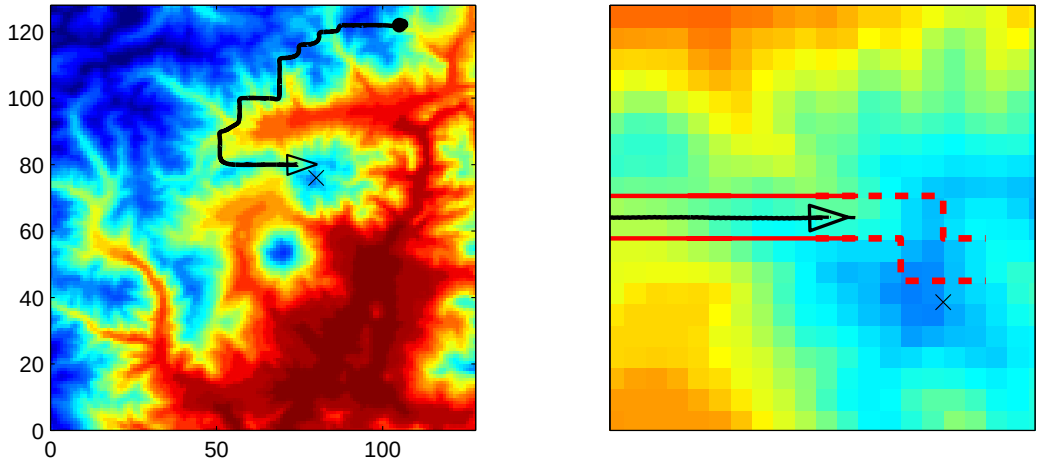
Figure 48: Simulation results of the hierarchical path control implementation. (cont'd)



(j) $t = 386.5$ [sec]

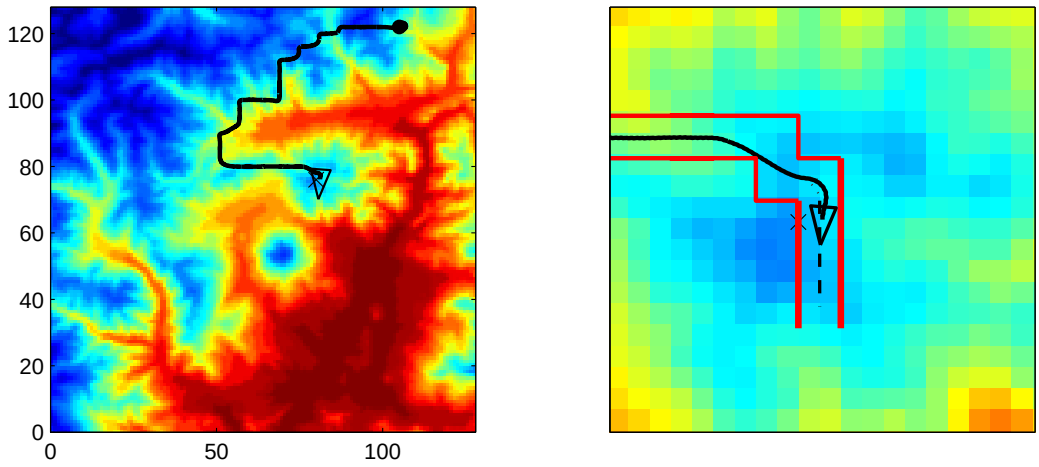


(k) $t = 429.0$ [sec]

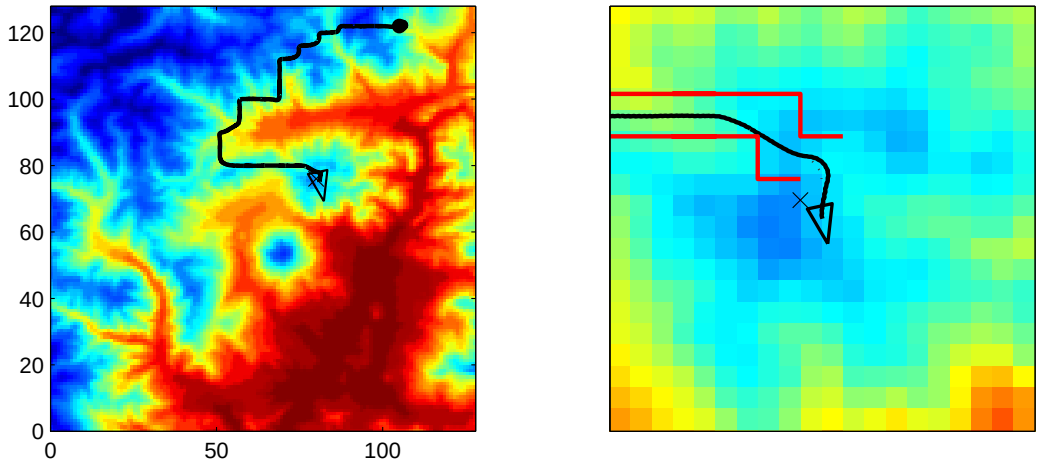


(l) $t = 492.5$ [sec]

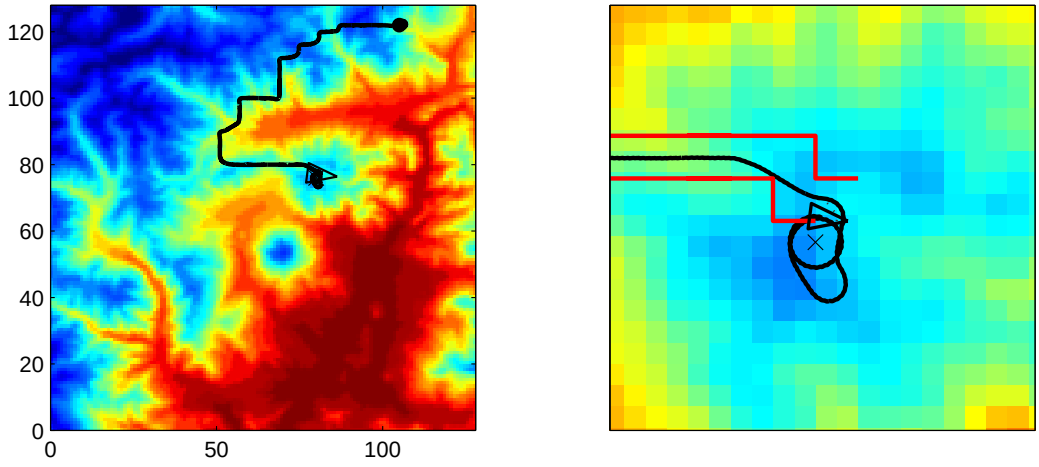
Figure 48: Simulation results of the hierarchical path control implementation. (cont'd)



(m) $t = 528.0$ [sec]



(n) $t = 532.0$ [sec]



(o) $t = 591.5$ [sec]

Figure 48: Simulation results of the hierarchical path control implementation. (cont'd)

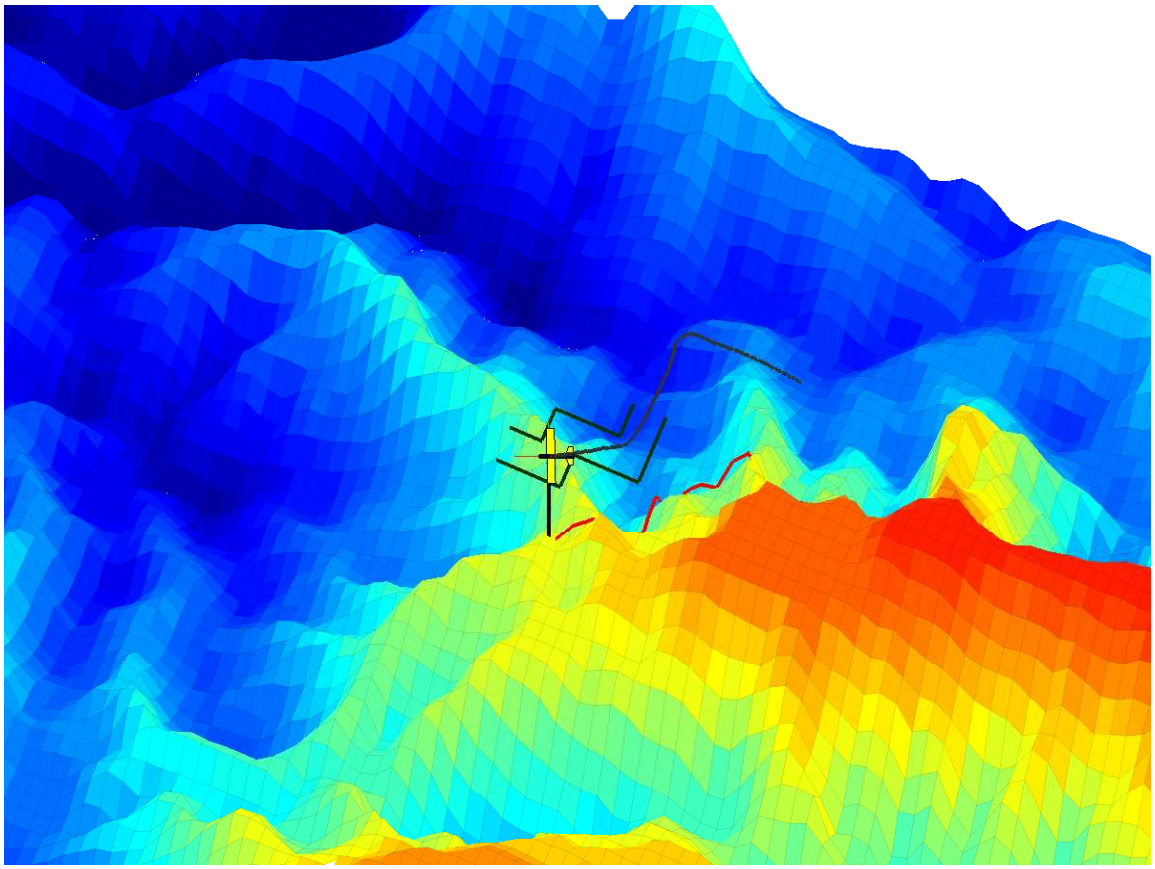


Figure 49: A 3D screen shot during the simulation. The ground track of the followed path is displayed showing that the UAV is avoiding the obstacles (for this case, it is the high elevation region).

CHAPTER VI

CONCLUSIONS AND FUTURE RESEARCH

6.1 Conclusions

Autonomous control for small UAVs imposes severe restrictions on the control algorithm development, stemming from the limitations imposed by the on-board hardware and the requirement for on-line implementation. In this thesis we have proposed a new hierarchical control scheme for the navigation and guidance of a small UAV for obstacle avoidance. The multi-stage control hierarchy for a complete path control algorithm is comprised of several control steps: Top-level path planning, mid-level path smoothing, and bottom-level path following controls. In each stage of the control hierarchy, the limitation of the on-board computational resources has been taken into account to come up with a practically feasible control solution. We have validated these developments in realistic non-trivial scenarios.

In Chapter 2 we proposed a multiresolution path planning algorithm. The algorithm computes at each step a multiresolution representation of the environment using the fast lifting wavelet transform. The main idea is to employ high resolution close to the agent (where is needed most), and a coarse resolution at large distances from the current location of the agent. It has been shown that the proposed multiresolution path planning algorithm provides an on-line path solution which is most reliable close to the agent, while ultimately reaching the goal. In addition, the connectivity relationship of the corresponding multiresolution cell decomposition can be computed directly from the the approximation and detail coefficients of the FLWT. The path planning algorithm is scalable and can be tailored to the available computational resources of the agent.

The on-line path smoothing algorithm incorporating the path templates is presented in Chapter 3. The path templates are comprised of a set of B-spline curves, which have been obtained from solving the off-line optimization problem subject to the channel constraints. The channel is closely related to the obstacle-free high resolution cells over the path sequence calculated from the high-level path planner. The obstacle avoidance is implicitly dealt with since each B-spline curve is constrained to stay inside the prescribed channel, thus avoiding obstacles outside the channel. By the affine invariance property of B-spline, each component in the B-spine path templates can be adapted to the discrete path sequence obtained from the high-level path planner. We have shown that the smooth reference path over the entire path can be calculated on-line by utilizing the path templates and path stitching scheme. The simulation results with the $\mathcal{D}^*$-lite path planning algorithm validates the effectiveness of the on-line path smoothing algorithm. This approach has the advantage of minimal on-line computational cost since most of computations are done off-line.

In Chapter 4 a nonlinear path following control law has been developed for a small fixed-wing UAV. The kinematic control law realizes cooperative path following so that the motion of a virtual target is controlled by an extra control input to help the convergence of the error variables. We applied the backstepping to derive the roll command for a fixed-wing UAV from the heading rate command of the kinematic control law. Furthermore, we applied parameter adaptation to compensate for the inaccurate time constant of the roll closed-loop dynamics. The proposed path following control algorithm is validated through a high-fidelity 6-DOF simulation of a fixed-wing UAV using a realistic sensor measurement, which verifies the applicability of the proposed algorithm to the actual UAV.

Finally, the complete hierarchical path control algorithm proposed in this thesis is validated thorough a high-fidelity hardware-in-the-loop simulation environment using the actual hardware platform. From the simulation results, it has been demonstrated

that the proposed hierarchical path control law has been successfully applied for path control of a small UAV equipped with an autopilot that has limited computational resources.

6.2 Future Research

In this section, several possible extensions of the work presented in this thesis are outlined.

6.2.1 Reusable graph structure

The proposed path planning algorithm involves calculating the multiresolution cell decomposition and the corresponding graph structure at each of iteration. Hence, the connectivity graph $\mathcal{G}(t)$ changes as the agent proceeds toward the goal. Subsequently, let $\mathsf{x} \in \mathcal{W}$ be a state (location) which corresponds to nodes of two distinct graphs as follows

$$\mathsf{cell}_{\mathcal{G}(t_i)}(v_m^i) = \mathsf{cell}_{\mathcal{G}(t_j)}(v_n^j), \quad i \neq j \tag{105}$$

By the respective $\mathcal{A}^*$ search on those graphs, the agent might be rendered to visit x at different time steps of t_i and t_j, $i \neq j$. As a result, a cyclic loop with respect to x is formed for the agent to repeat this pathological loop, while never reaching the goal. Although it has been presented that maintaining a visited set might be a means of avoiding such pathological situations[142], it turns out to be a trial-and-error scheme is not a systemical approach. Rather, suppose that we could employ a unified graph structure over the entire iteration, which retains the information from the previous search. Similar to the $\mathcal{D}^*$-lite path planning algorithm, the incremental search over the graph by reusing the previous information results in not only overcoming the pathological situation but also reducing the computational time. In contrast to $\mathcal{D}^*$ or $\mathcal{D}^*$-lite algorithms where a uniform graph structure is employed, a challenge lies in building the unified graph structure from a multiresolution cell decomposition. Specifically, it includes a dynamic, multiresolution scheme for constructing the graph

connectivity between nodes at different levels. The unified graph structure will evolve itself as the agent moves, while updating nodes and edges associated with the multiresolution cell decomposition from the FLWT. If this is the case, we might be able to adapt the proposed path planning algorithm to an incremental search algorithm, hence taking advantages of both the efficient multiresolution connectivity (due to the FLWT) and the fast computation (due to the incremental search by using the previous information).

6.2.2 Kinodynamically feasible trajectory generation using B-splines

In this thesis, we utilized a B-spline representation of the reference path for the path smoothing purpose. In general, B-spline curves provide only information in spatial terms without taking into account evolution in terms of time, thus they are not useful to represent state references that are dependent on time. This aspect imposes severe restrictions on designing control algorithms for time critical applications. On the other hand, several authors in the literature [147, 39] proposed to adapt B-spline to explicitly deal with the time variable to design a smooth trajectory subject to the kinematic constraints of the vehicles. Let us consider the following unicycle-like kinematic equations of motion,

$$\dot{x} = v\cos\psi, \tag{106a}$$

$$\dot{y} = v\sin\psi, \tag{106b}$$

$$\dot{\psi} = \omega, \tag{106c}$$

Suppose the reference trajectory can be represented by a B-spline curve in terms of the curve parameter u,

$$\mathbf{r} = \big(x_d(u), y_d(u)\big). \tag{107}$$

It follows from the basic theory of differential geometry for curves[25] that the speed of the agent is determined by,

$$v(u) = \sqrt{\left(\frac{\mathrm{d}x_d}{\mathrm{d}u}\right)^2 + \left(\frac{\mathrm{d}y_d}{\mathrm{d}u}\right)^2}, \tag{108}$$

and the curvature is calculated as

$$\kappa(u) = \frac{x_d'' y_d' - y_d'' x_d'}{(x_d'^2 + y_d'^2)^{3/2}}, \tag{109}$$

where $(\cdot)'$ and $(\cdot)''$ denote the first and second order derivative with respect to u, respectively. The angular velocity can be determined using Eqs. (106) as

$$\omega = \frac{\ddot{y}\dot{x} - \dot{y}\ddot{x}}{\dot{x}^2 + \dot{y}^2}, \tag{110}$$

and can be represented using Eqs (108) and (109) as follows,

$$\omega = \kappa(u)v(u) \tag{111}$$

The kinematic constraints of the vehicles are given by

$$|\omega| \leq \omega_{\max}, \tag{112}$$

and $v \in [v_{\min}, v_{\max}]$, using Eqs. (111) and (112), we get

$$v_{\min} \leq v \leq \left(\frac{\omega_{\max}}{\kappa}\right). \tag{113}$$

Subsequently, an off-line optimization problem can then be formulated taking into consideration the kinematic constraints shown in Eqs. (112) and (113). The path templates which explicitly deal with the kinodynamical constraints such as the bounded flight speed or the minimum turning radius are constructed from a set of off-line optimization problems.

6.2.3 Trajectory tracking controller design using differential flatness

In this thesis, we proposed the path following control law for tracking a reference path. Although the path following control algorithm was successfully implemented

to follow a smooth reference path, no explicit consideration of vehicle dynamics and the kinematic constraints of a fixed-wing UAV was given. Furthermore, a non-zero constant speed of the UAV was assumed, hence it is not possible for the UAV to track the time-stamped trajectory using the proposed path following controller. In order to take into account the trajectory tracking capability of the UAV in conjunction with the vehicle dynamics, we consider a simplified kinematic model in the two dimensional plane as follows,

$$\dot{x} = u_1 \cos\chi, \tag{114a}$$

$$\dot{y} = u_1 \sin\chi, \tag{114b}$$

$$\dot{\chi} = u_2, \tag{114c}$$

where (x, y) denotes the inertial position and χ is the inertial heading angle. The control inputs $\mathbf{u} = [u_1 \; u_2]^\mathsf{T}$ consist of the forward flight speed and the heading rate of the UAV, respectively. Given a reference trajectory, we design a control law for the flight speed and the heading rate commands. To this end, we let $\mathbf{z} = [z_1 \; z_2]^\mathsf{T}$ be an output of the system,

$$z_1 = x, \qquad z_2 = y. \tag{115}$$

It follows from Eqs. (114a) and (114b) that

$$\tan\chi = \frac{\dot{y}}{\dot{x}}, \tag{116}$$

which shows that the system state $\mathbf{x} = [x \; y \; \chi]^\mathsf{T}$ is represented by the output as follows,

$$\mathbf{x} = \begin{bmatrix} z_1 \\ z_2 \\ \tan^{-1}\left(\frac{\dot{z}_2}{\dot{z}_1}\right) \end{bmatrix} \tag{117}$$

The vehicle forward flight speed is obtained from Eqs. (114a) and (114b), which yields the control input u_1 expressed in terms of the output as follows,

$$u_1 = \sqrt{\dot{x}^2 + \dot{y}^2} \tag{118}$$

By differentiating Eq. (116) with respect to time and using trigonometry, we obtain the expression for u_2 in terms of the output

$$u_2 = \frac{\ddot{y}\dot{x} - \dot{y}\ddot{x}}{\dot{x}^2 + \dot{y}^2}. \tag{119}$$

The control vector $\mathbf{u}$ is represented by the output as follows,

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} \sqrt{\dot{z}_1^2 + \dot{z}_2^2} \\ \frac{\ddot{z}_2\dot{z}_1 - \dot{z}_2\ddot{z}_1}{\dot{z}_1^2 + \dot{z}_2^2} \end{bmatrix} \tag{120}$$

Therefore, the system in Eq. (114) is proved differentially flat in conjunction with the flat output $\mathbf{z} = [x\ y]^\mathsf{T}$, since each state and control input can be expressed in terms of the flat output and its higher derivatives,

$$\mathbf{x} = \mathcal{X}(z_1, z_2, \dot{z}_1, \dot{z}_2), \tag{121a}$$

$$\mathbf{u} = \mathcal{W}(z_1, z_2, \dot{z}_1, \dot{z}_2, \ddot{z}_1, \ddot{z}_2) \tag{121b}$$

Now, given a desired trajectory in terms of $\mathbf{x}(t)$, $\dot{\mathbf{x}}_d(t)$, $\ddot{\mathbf{x}}_d(t)$, we can design a trajectory tracking control law in the flat space $\mathcal{E} = \{\mathcal{X}, \mathcal{W}\}$, explicitly dealing with the input constraints of the forward flight speed, as specified by

$$v_{\min} \leq u_1 \leq v_{\max}, \tag{122}$$

and the bounded heading rate as follows,

$$|u_2| \leq \omega_{\max}. \tag{123}$$

6.2.4 Path planning in three dimensions

The proposed path planning algorithm is based on the assumption that the UAV is restricted to navigate through a two dimensional environment, that is, the UAV flies at constant altitude while avoiding in-plane obstacles. On the other hand, by allowing the UAV to change altitude during the mission, we can take the advantage of three dimensional maneuvers of the UAV to plan a realistic three dimensional path

that avoids 3D obstacles effectively. To this end, without loss of generality, we let $\mathcal{W} = [0,1] \times [0,1] \times [0,1]$. We assume that we are given a 3D function $\mathsf{RM} : \mathcal{W} \mapsto \mathcal{M}$ that represent the risk measure at the location $\mathsf{x} = (x, y, z)$. This three dimensional function is tailored to capture not only the aspect of 3D risk measure depending on the operational altitude but also the attainable maneuverability of the UAV between distinct altitudes. We apply the multidimensional FLWT on this function, which results in the the following decomposition,

$$\mathsf{RM}(x,y,z) = \sum_{k,\ell,m=0}^{2^J-1} a_{J,k,\ell,m}\Phi_{J,k,\ell,m}(x,y,z) + \sum_{i=1}^{8}\sum_{j=J}^{N-1}\sum_{k,\ell,m=0}^{2^j-1} d^i_{j,k,\ell,m}\Psi^i_{j,k,\ell,m}(x,y,z), \tag{124}$$

where $\Phi_{J,k,\ell,m}(x,y,z)$ and $\Psi^i_{j,k,\ell,m}(x,y,z)$, $i = 1, \cdots, 8$ are families of function, which are derived from a linear combination of both the scaling function $\phi(\cdot)$ and the wavelet function $\psi(\cdot)$ in each coordinate. Subsequently, we are able to obtain the multiresolution, three dimensional cell decomposition of $\mathcal{W}$, which is further associated with a topological graph structure with a connectivity information. It is then imperative that we extend the approach discussed in Chapter 2 in order to construct the connectivity relationship of the graph by utilizing the corresponding approximation and detail coefficients of the FLWT. Consequently, the rest of 3D path planning collapses to finding an optimal sequence of cells in the graph structure in conjunction with standard graph search methods such as $\mathcal{A}^*$ or Dijkstra's algorithm.

APPENDIX A

UAV AVIONICS DESCRIPTION

A.1 System Architecture

The overall architecture of the UAV system is shown in Fig. 50. The main subsystems are the autopilot, the ground station, and the interconnection between the two. The on-board autopilot is equipped with a micro-controller, sensors and actuators, along with the communication devices that allow full functionality for autonomous control. The micro-controller provides data acquisition, processing, and communication with the ground station. It also runs the main control software. Table 6 shows the detail specification of the micro-controller. The on-board sensors include angular rate sensors for three axes, accelerometers for three axes, a three-axis magnetic compass, a GPS sensor, an engine RPM sensor, absolute and differential pressure sensor, battery voltage sensor, and temperature sensor.

Table 6: Specifications of the Rabbit RCM-3400 micro-controller module.

Parameters	Values
Clock speed	29.4MHz
Programmable memory	512KByte
Data memory	512KByte
Analog Inputs	Eight channels Single-Ended, 11bit resolution
Serial ports	Six configurable asynchronous/SPI SDLC
Timers	Ten independent 8bit timers
Pulse-Width Modulators	four independent 10bit free-running counters
Power consumption	Max. 100mA at 3.3V operation
Size	$1.37'' \times 1.16'' \times 0.31''$

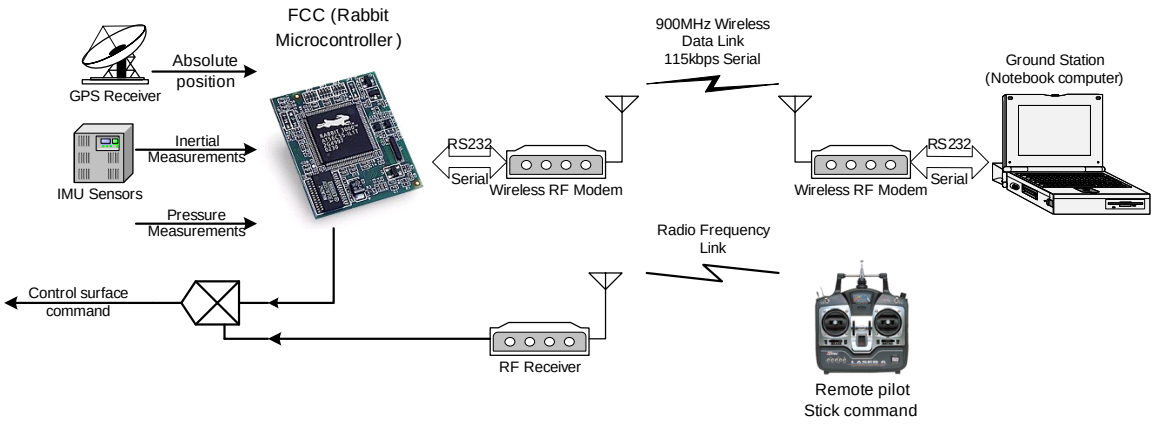


Figure 50: System architecture of the UAV test-bed.

A.2 Autopilot

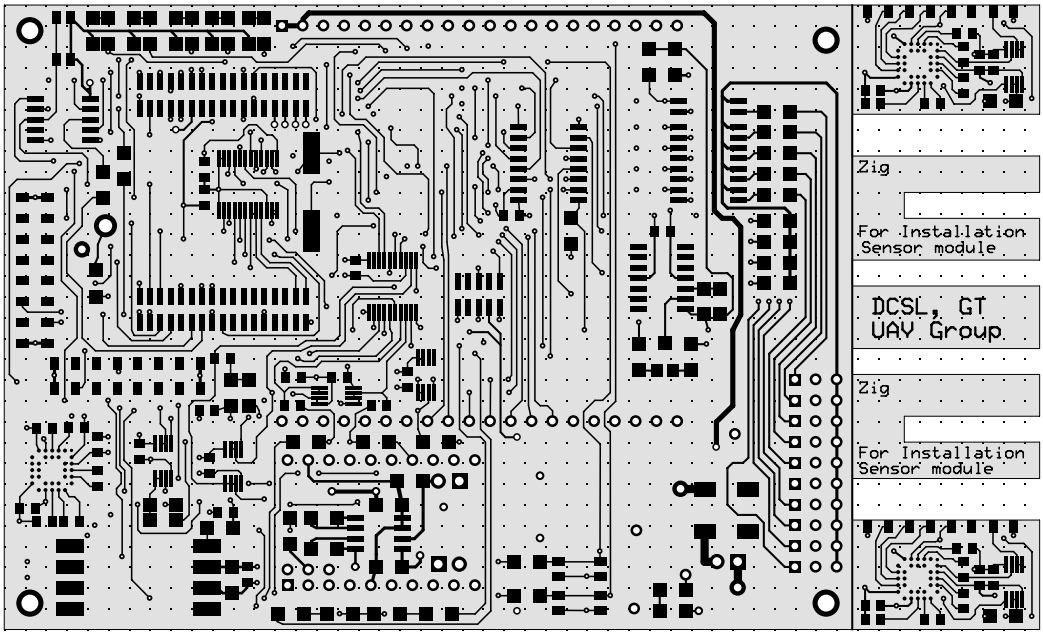
The autopilot box contains all hardware components, such as the micro-controller, all sensor ICs, signal conditioning circuitry, data acquisition devices, and the wireless modem board.

A.2.1 Sensor Board

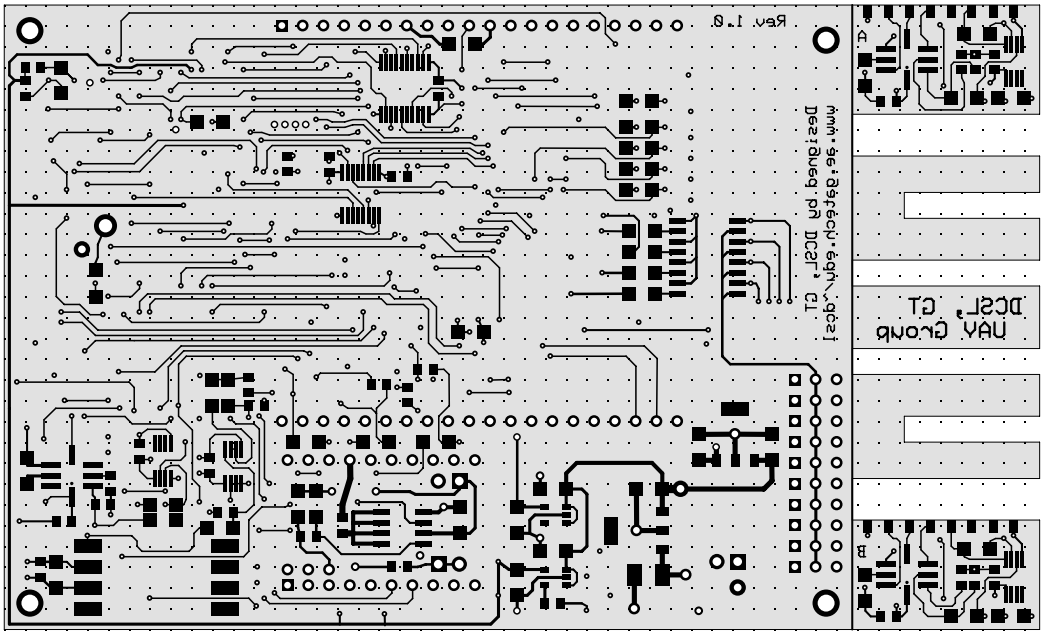
The microprocessor, sensors and associated electronics were integrated on a custom-designed and fabricated four-layer 5″ by 3″ printed circuit board (PCB). Figure 51 shows the detail layout of the four-layer sensor board for the top and bottom layers. The sensor board is equipped with three single-chip rate gyros, three two-axis accelerometers, a three-axis magnetometer, two pressure sensors, and a GPS receiver interfacing to the micro-controller module. It also includes the power regulating circuitry that supplies power for all electronic components. Figure 52 shows the functional diagram of the sensor board and Fig. 53 shows the top and side views of the completed sensor board with all components assembled.

A.2.2 Inertial Sensors

Three ADXRS150 angular rate sensors from Analog Devices provide three-axis body-fixed angular rate measurements. Measurements of linear accelerations in all three



(a) Top layer design



(b) Bottom layer design

Figure 51: Sensor board design layout. The board is 5″ by 3″ printed circuit board (PCB). Four layers include the power plane and the ground plane (Not shown above).

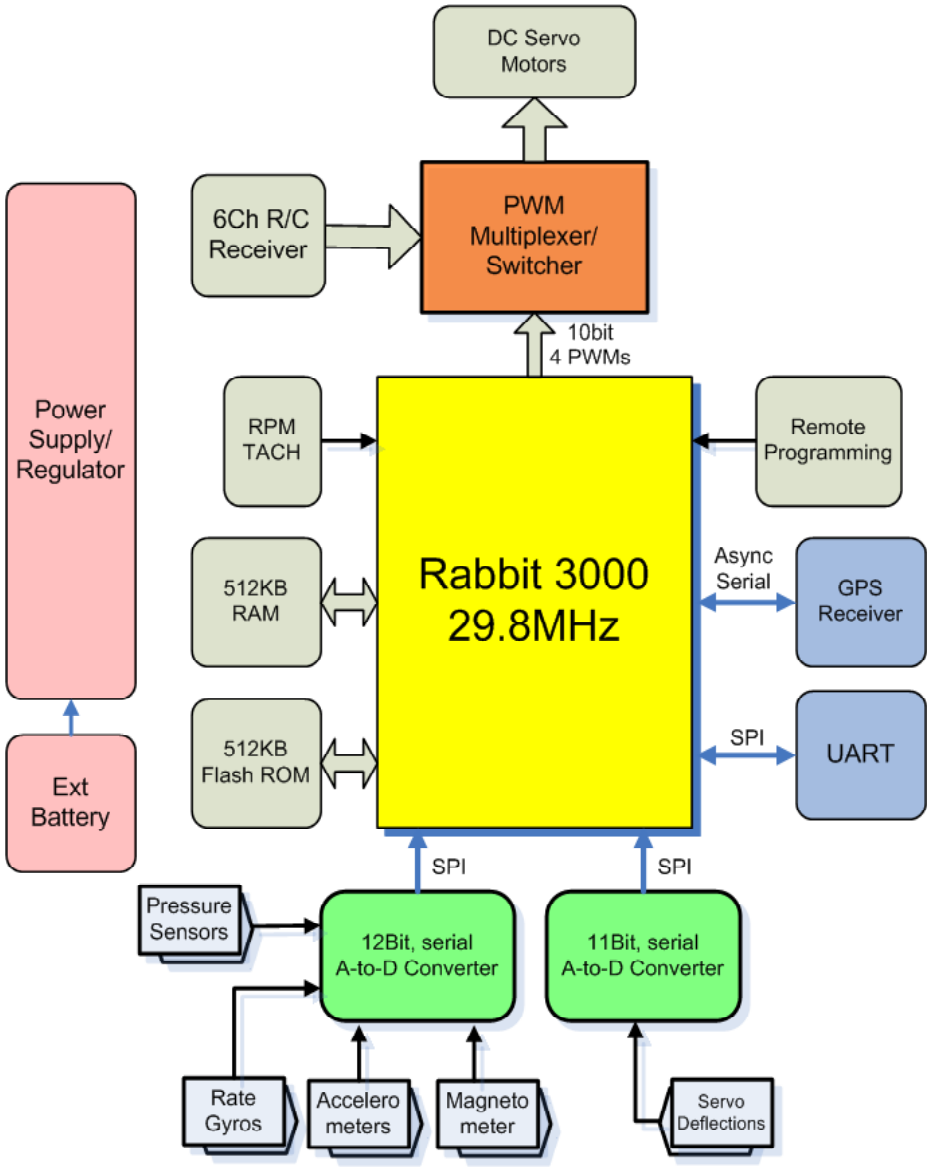
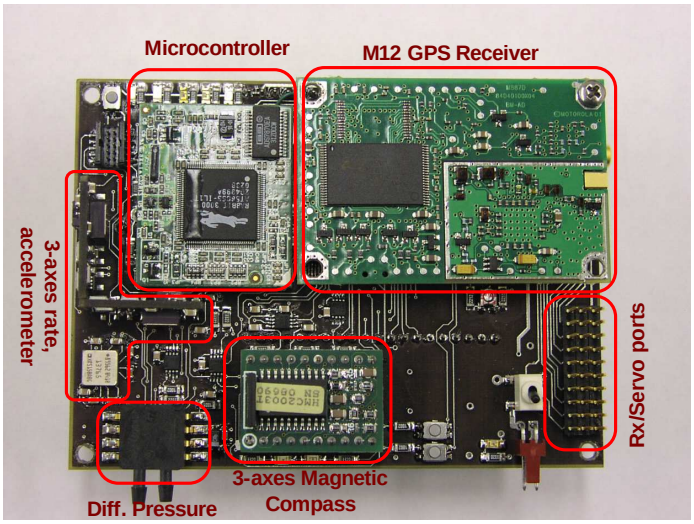
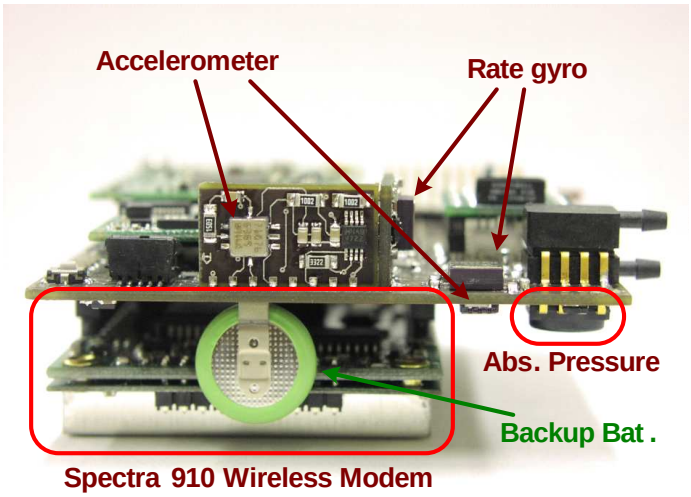


Figure 52: Sensor board functional block diagram.



(a) Top view



(b) Side view

Figure 53: Assembled autopilot hardware.

Table 7: Inertial sensors specifications

Parameters	Values	Remarks
MEMS angular rate sensor, Analog Device ADXRS150		
Dynamic range	± 150 °/sec	
Sensitivity	12.5±10% mV/°/sec	
Sensitivity nonlinearity	0.1% FS	from the best fit line
Rate Noise Density	0.05 °/sec/$\sqrt{\text{Hz}}$	
Bandwidth	DC to 2 kHz	
MEMS linear acceleration sensor, Analog Device ADXL202		
Dynamic range	± 2 g	1 g = 9.81 m/sec^2
Sensitivity	312 mV/g	
Sensitivity nonlinearity	0.2% FS	from the best fit line
Acceleration Noise Density	200 μg/$\sqrt{\text{Hz}}$	
3dB Bandwidth	DC to 6 kHz	Maximum
3-axis magnetometer, Honeywell HMC2003 module		
Dynamic range	± 2 gauss	
Sensitivity nonlinearity	0.5% FS	from the best fit line
Resolution	40 μgauss	
Bandwidth	1 kHz	

axes are provided by three ADXL202 dual-axis accelerometers from Analog Devices. A three-axis magnetometer module HMC2003 from Honeywell Solid State Electronics Center (SSEC) is employed to obtain absolute orientation angles with respect to the Earth by sensing Earth's magnetic field. Table 7 displays the detail specification of the inertial sensors. A GPS receiver (Motorola OnCore M12) has been used to provide absolute position of the UAV in the Earth-fixed Earth-centered (EFEC) coordinate frame. The output data of the GPS sensor is directly connected to a serial port on the micro-controller using the standard NMEA format or Motorola's native binary format at a rate of 1 Hz.

A.2.3 Other Sensors

An MPXV5004D differential pressure sensor that can measure pressures up to 3.92 kPa was used in conjunction with a custom-made pitot-tube, attached under the left

Table 8: Specifications of the autopilot sensors.

Sensors	Range	Resolution	1-σ noise
Accelerometer	± 2 g	0.004 g	0.025 g
Rate gyro	± 150 °/sec	0.1 °/sec	0.4 °/sec
Magnetometer	± 2 gauss	1.22 mgauss	4 mgauss
Absolute pressure	Above sea level	2.75 m	3 m
Differential pressure	79.2 m/sec	1.40 m/sec	1.5 m/sec
Servo Position	± 60 deg	0.5 deg	

wing to obtain the airspeed. The altitude of the airplane is obtained from the pressure differential referred by the ground level, as it is measured during flight via an MPXAZ4115 pressure sensor.

Engine thrust can be approximately calculated from the knowledge of the engine RPM. The engine RPM is measured by attaching two very small magnets ($1/4''$ diameter) on the back plate of the spinner, and by using a non-contact hall-effect sensor that is fixed on the cowling of the airplane. The hall sensor generates electrical pulses whenever the magnet passes in front of it as the propeller spins. By measuring the time interval between each pulse the micro-controller can calculate the engine/propeller speed with a resolution of 1 rpm.

The airplane's control surfaces are actuated with the help of a series of DC servo motors. To obtain command input information for model identification purposes we should have accurate knowledge of the deflection angles of all the aerodynamic surfaces (elevator, rudder, ailerons) as well as the throttle setting. These are obtained by measuring the voltage of the potentiometer connected by mechanical link to each DC servo motor. This approach allows to measure the control surfaces deflections with a resolution of 0.5 [deg]. Details from the calibration of potentiometers for accurate deflection angles are given in Section A.4.3.

Table 8 summarizes the specifications, operational range, resolution, and noise performance of the autopilot sensors.

A.2.4 Communication Modem

The UAV has two main remote communication links (a third, independent link which is used to provide live video is not described here). The first link (RF band) uses the standard communication channel between the remote control (Futaba) and the airplane. The second link provides the main data communication backbone between the airplane and the ground station (see Section A.3). These two links are kept completely separate for safety reasons. A Spectra 910 wireless modem was utilized to set up a data communication link between the autopilot and the ground station. The Spectra 910 operates in the license-exempt 900 MHz frequency band utilizing frequency-hopping spread-spectrum, and is capable of providing reliable wireless data transfer up to distance of 25 miles LOS under ideal conditions (at maximum transmitting power). The interface with the micro-controller is achieved via a standard RS-232 serial communication at a maximum baud rate of 115200 bps.

A.2.5 Servo Motor Control

The micro-controller has four independent PWM outputs that generate reference command to the motors in pulse form with a varying pulse width according to the desired position. The frequency of the pulse was identified by 75 [Hz], and the duty-ratio (the ratio between [On] time versus [Off] time of the pulse) changes from 5% to 15% for the maximum allowable positions in positive and negative direction, respectively.

To achieve seamless integration (as well as switching back and forth) between autopilot and remote control action the native signal commands from the R/C receiver are merged with the PWM generated output from the micro-controller using a multiplexer. Switching of the multiplexer is being toggled by the remote pilot using a switch on the Futaba transmitter.

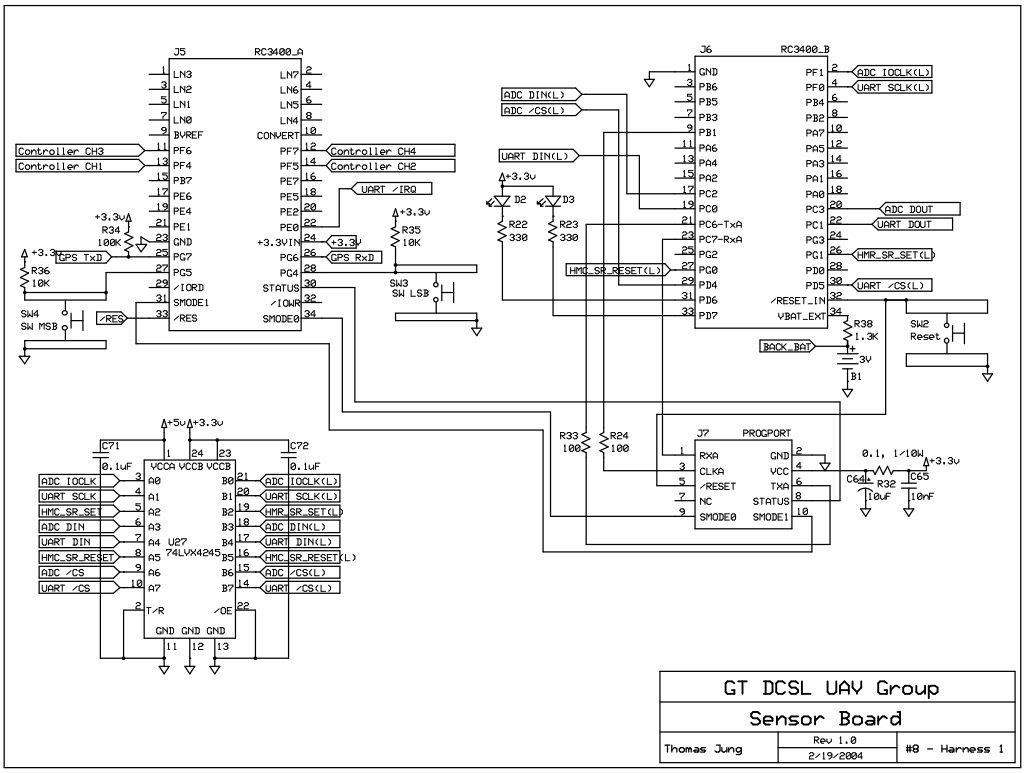
A.2.6 Schematics

In this section, detail electronic designs of the autopilot are presented. The RCM-3400 micro-controller module is interfaced through 64 pins (34 pins×2), as shown in Fig. 54(a). The micro-controller has seven ports (PA-PG) which are associated with various functionality of the micro-controller, allowing the interface to external peripherals. The autopilot adopts three independent drop-down power regulators for supplying 3.0V, 3.3V, and 5.0V, as shown in Fig. 54(b). Figures 55-56 show the detail interface circuits for the inertial sensors. The interface circuits for two pressure sensors are shown in Fig. 57(a), and the serial interface for both the M12 GPS module and the Spectra 910 is depicted in Fig. 57(b).

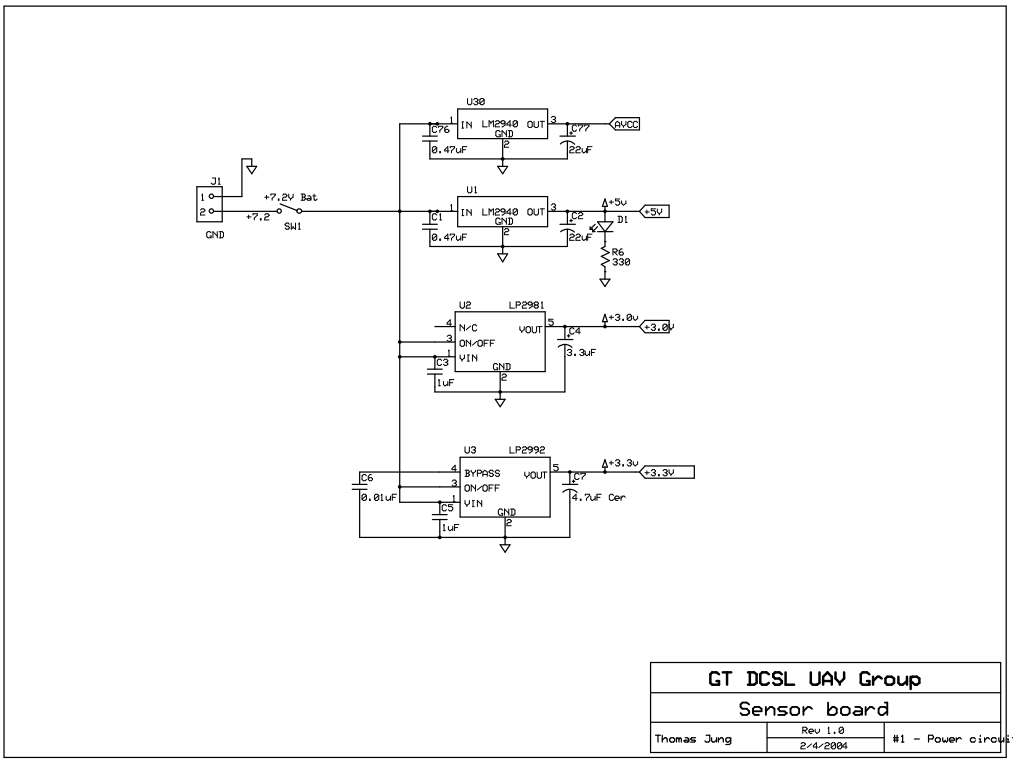
Figure 58(b) shows the schematic diagram for the control switching capability between the autopilot and the remote pilot stick command using a multiplexer.

A.3 Ground Station

The ground station consists of a laptop computer with a wireless communication modem. The laptop runs a Windows-based Graphical User Interface (GUI) program developed in-house, shown in Fig. 59. The ground station program provides real-time flight information by displaying all relevant system parameters, sensor readings, etc. A graphical dashboard representing a virtual horizon, altitude and speed has been adopted in the GUI panel to show all information graphically. A map of the area of the UAV's operation can be overlaid on the map panel in order to provide the user with the navigational details of the UAV via GPS data. The ground station program is also capable of coordinating the autonomous flight of the airplane by providing high level navigation control command via way-points on the map specified by the operator.

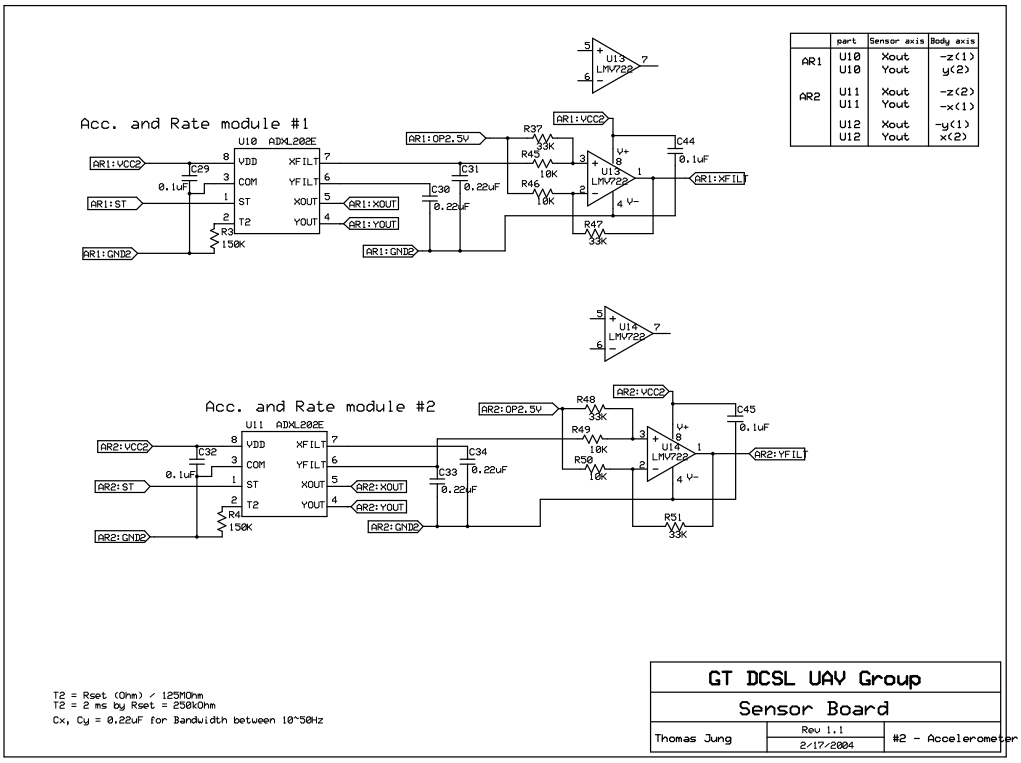


(a) Micro-controller interface circuits.

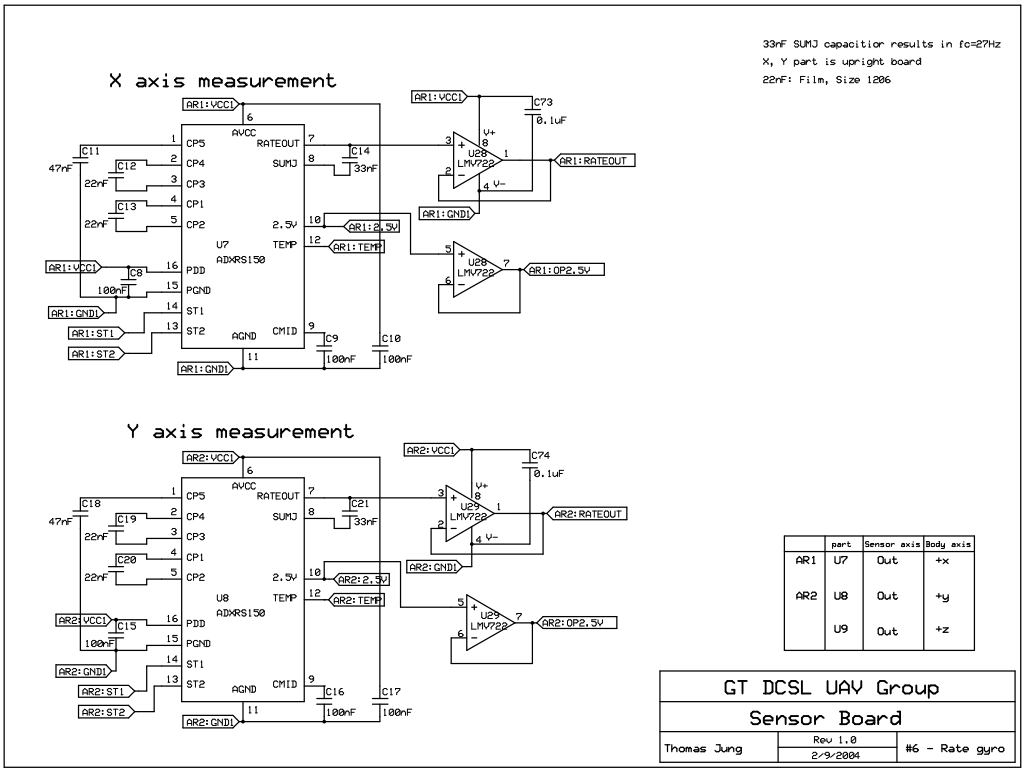


(b) Power regulators circuits.

Figure 54: Schematic diagrams of the designed autopilot: Power circuit and the micro-controller interface.

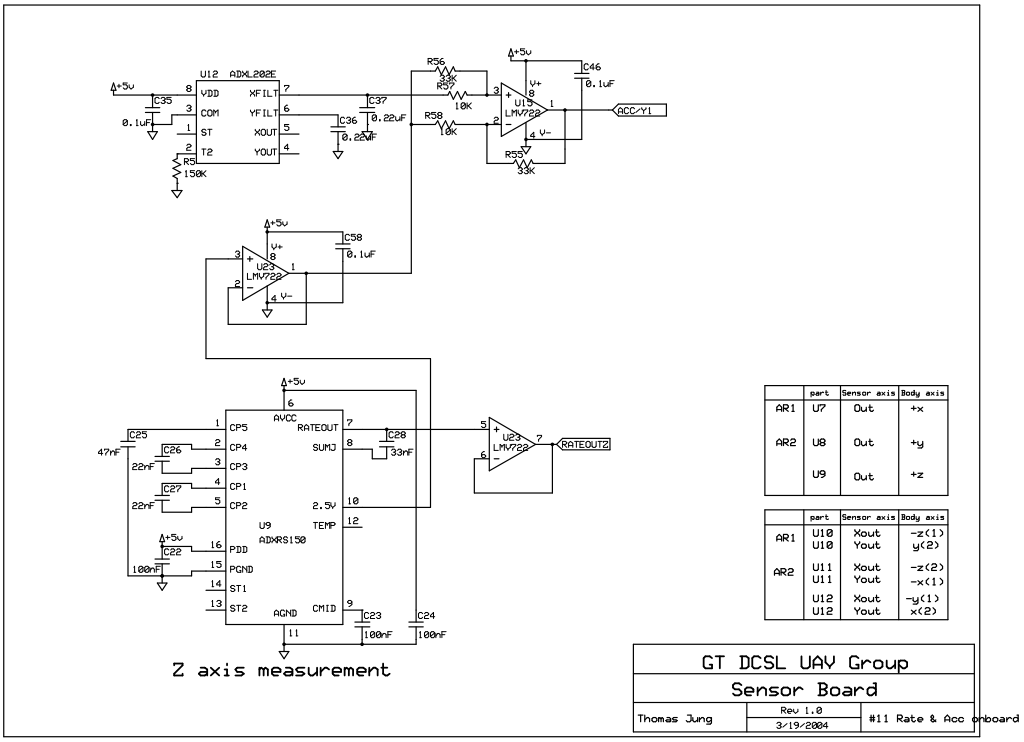


(a) ADXL202 interface circuits.

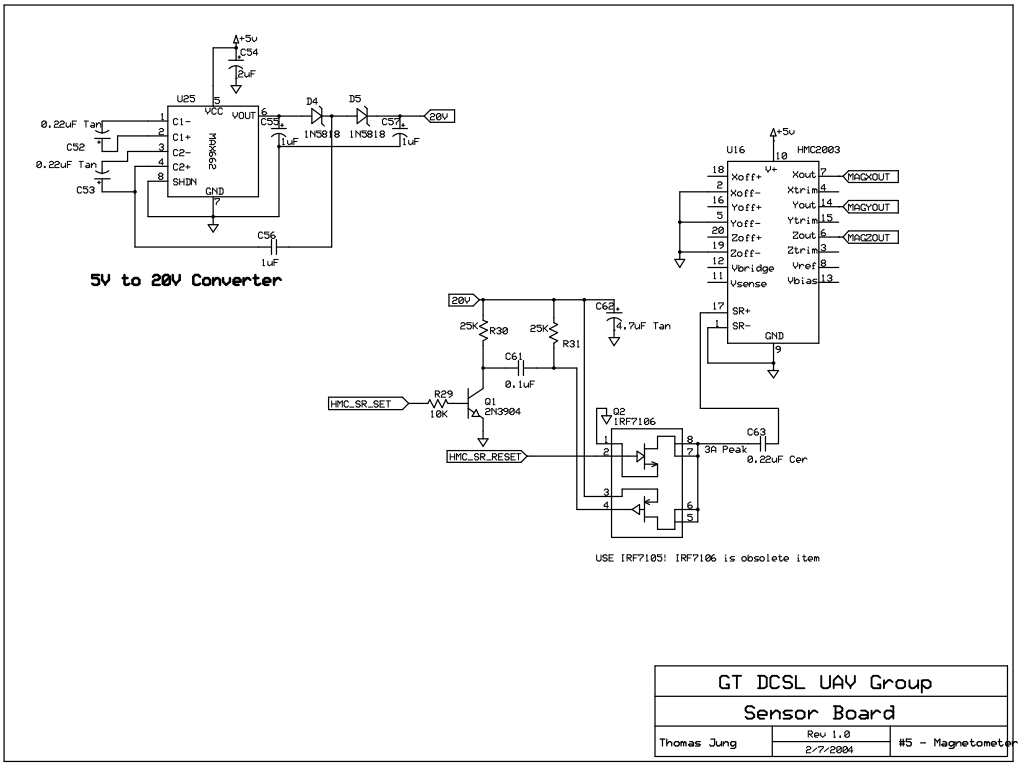


(b) ADXRS150 interface circuits.

Figure 55: Schematic diagrams of the designed autopilot: The rate sensors and the accelerometers interface.

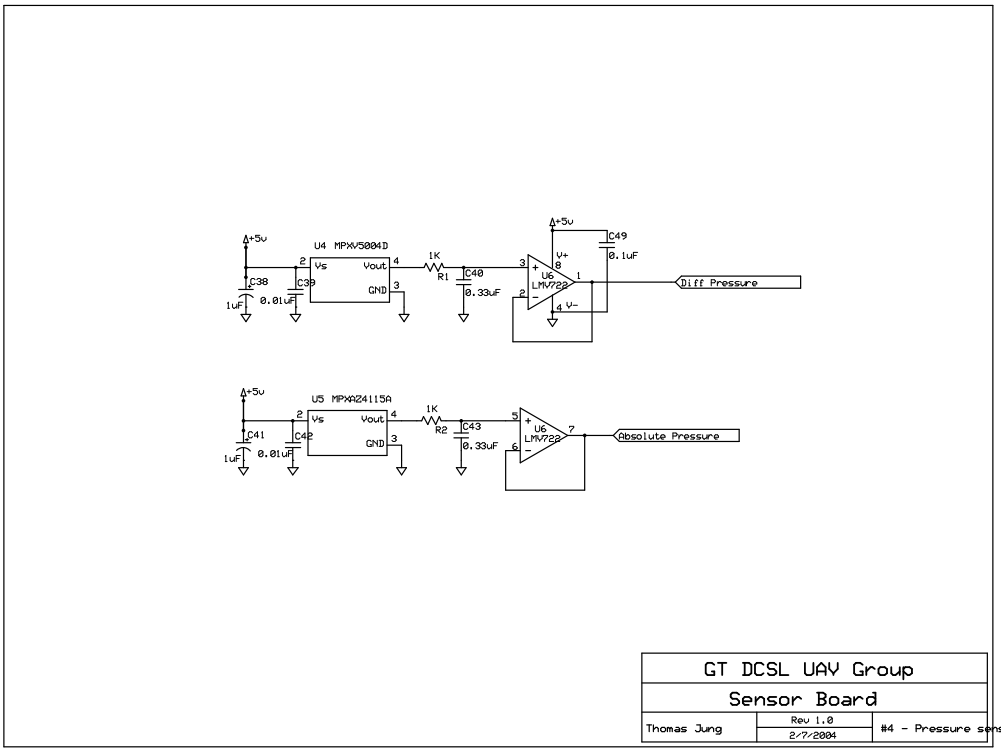


(a) ADXL202 and ADXRS150 interface circuits.

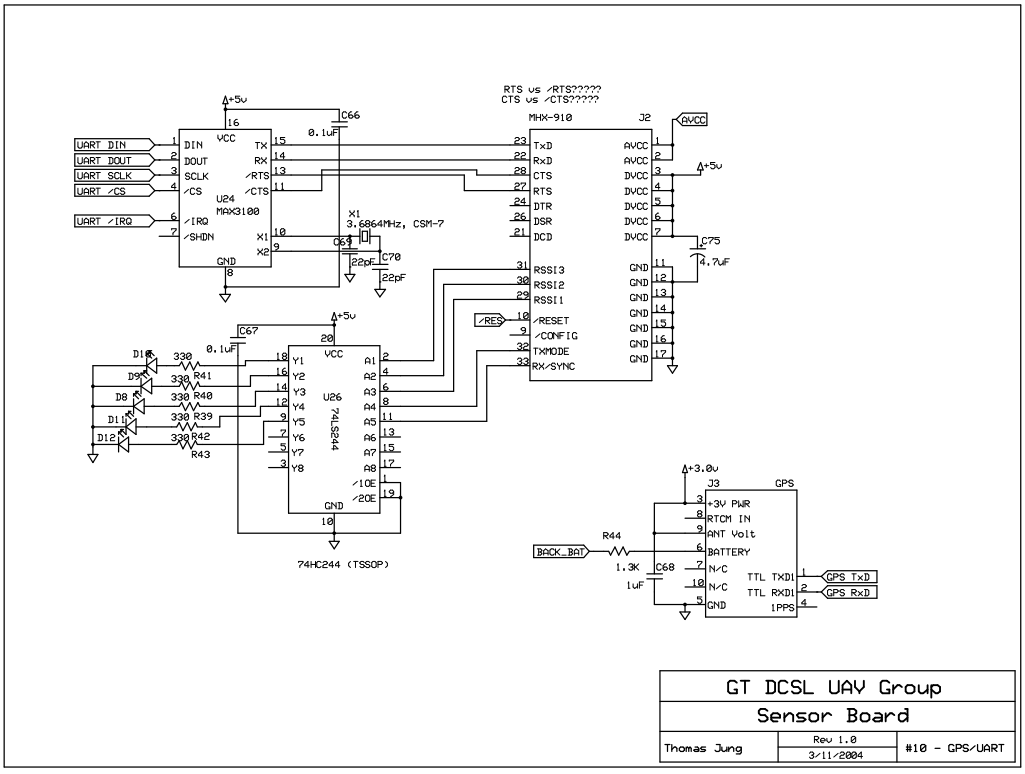


(b) HMC2003 interface circuits.

Figure 56: Schematic diagrams of the designed autopilot: Z-axis rate and acceleration, 3-axis magnetometer interface.

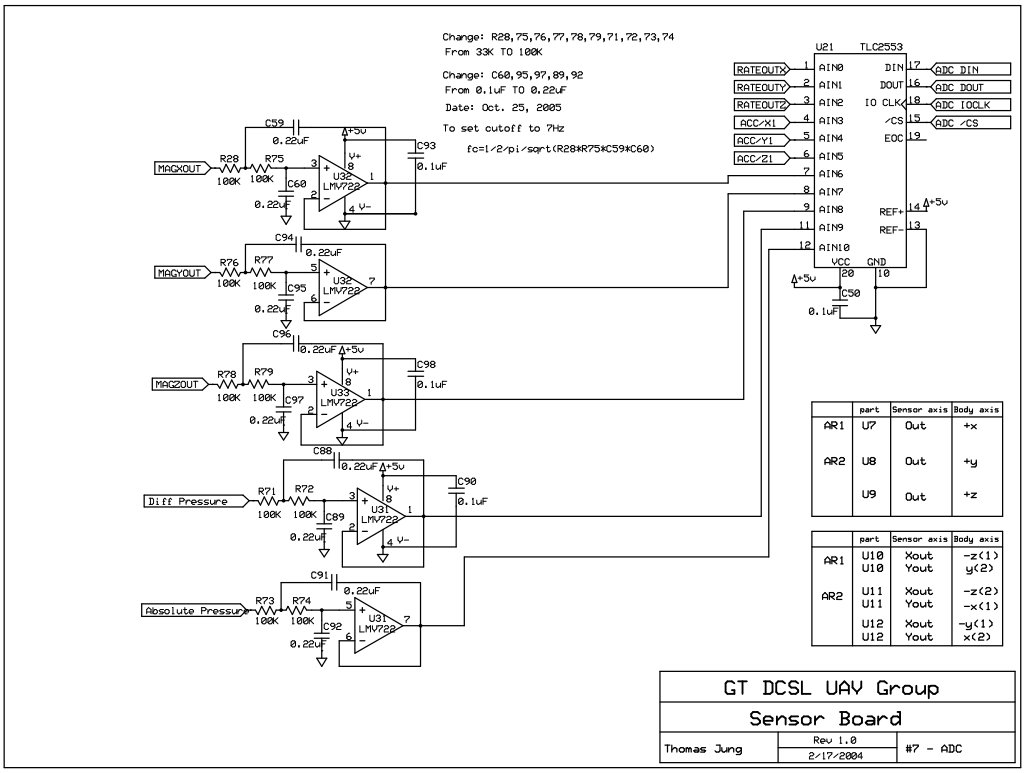


(a) MPXV5004D and MPXAZ4115 pressure sensor interface circuits.

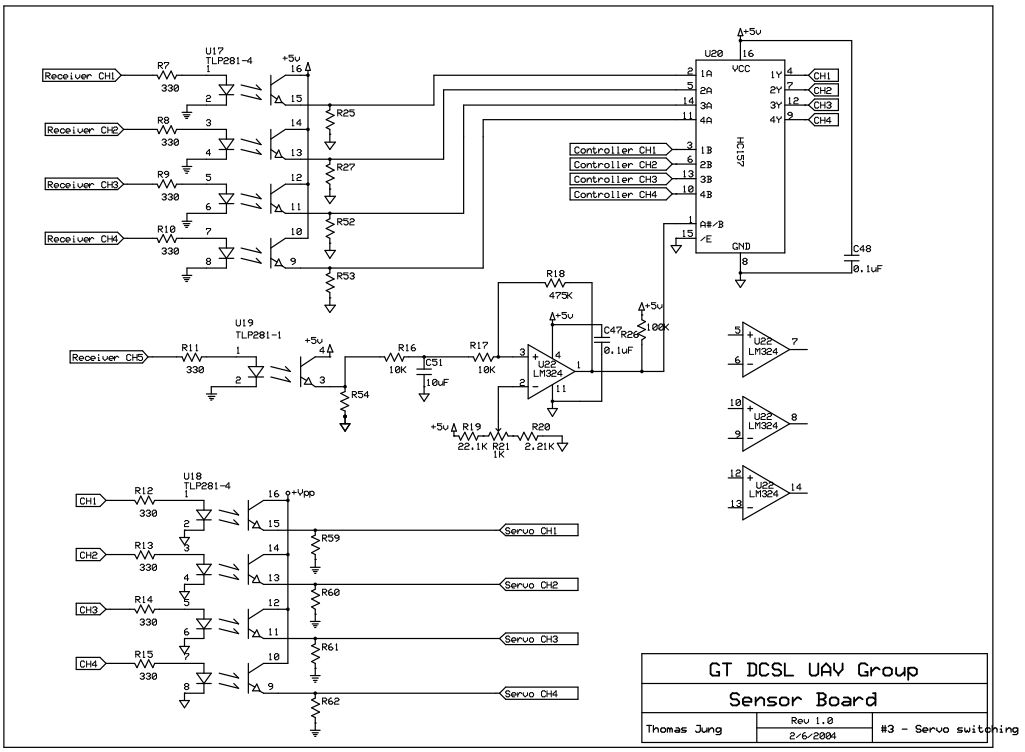


(b) M12 oncore GPS receiver and the Spectra 910 interface circuits.

Figure 57: Schematic diagrams of the designed autopilot: Pressure sensors and the GPS interface.



(a) Analog-to-Digital converter interface circuits.



(b) Servo switching circuits.

Figure 58: Schematic diagrams of the designed autopilot: A-to-D converter and the servo switching interface.

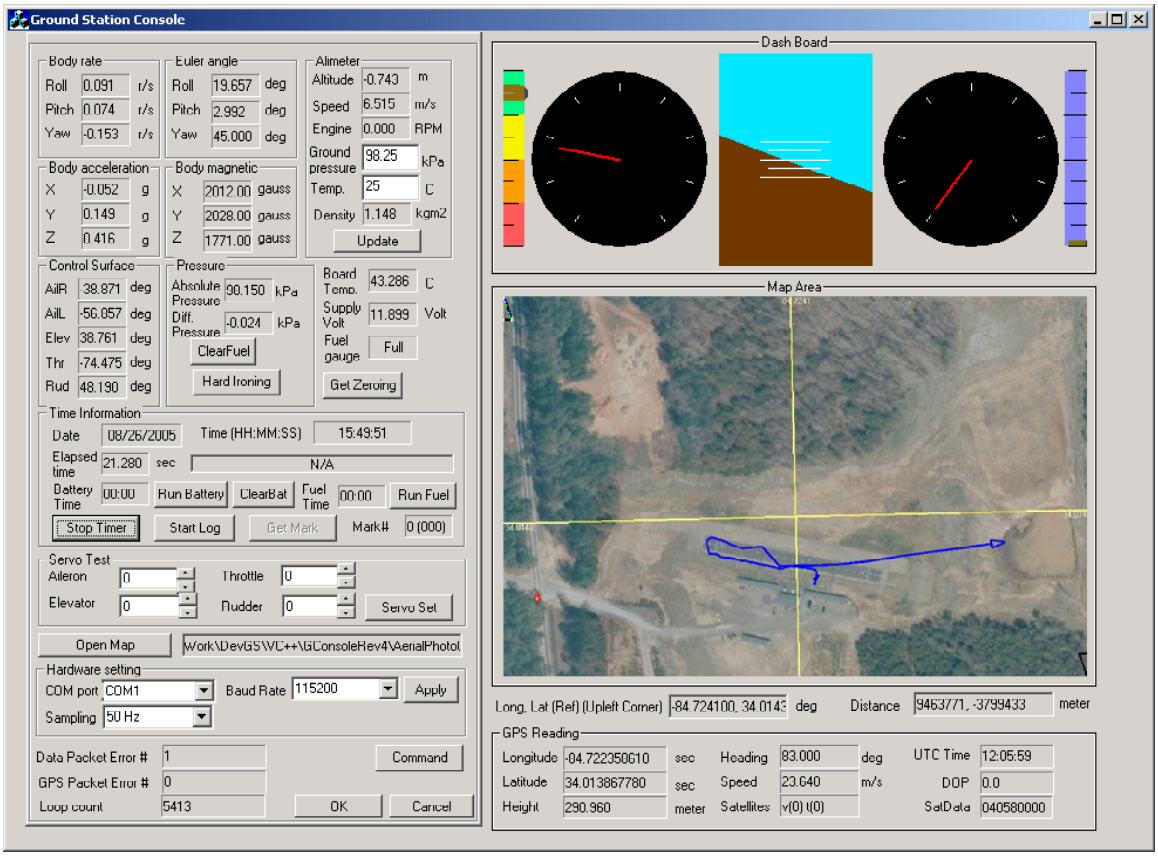


Figure 59: The ground station GUI program.

A.4 Hardware Evaluation/Calibration

A.4.1 Inertial Sensor Calibration

A static test was performed to determine the initial biases and the static noise level of each sensor. The sensor outputs were measured while the autopilot was completely stationary, and by performing a statistical analysis on the recorded data over time, the initial biases and 1-σ noise levels were obtained. The static noise characteristics of all sensor correlated favorably with the specification provided by the sensor manufacturer, and summarized in Table 8.

The actual scale factors for each accelerometer can be found by taking two measurements with the accelerometer's measurement axis pointed directly towards (+1g) or opposite (-1g) to the Earth. The scale factors can then be found from the difference of two measurements factored by the known gravity change (2g).

The scale factors of the angular velocity sensors were found by placing the autopilot on a three-axis rotational platform [57]. The platform is equipped with a high-performance angular rate gyro and an inertial measurement unit (IMU) which is capable of measuring angular velocities and linear accelerations in all three-axes with an accuracy better than 0.03 [deg/sec], and 0.001 [g], respectively. After the autopilot was securely mounted on the platform, the platform was set in motion while both signals from the autopilot and from the high-performance platform sensors were recorded. The rate sensor outputs were then compared, and a least square fit was employed to find the best scale factor of the autopilot rate sensors. Figure 60 shows the result from this approach. In addition, Fig. 61 shows the validation of estimated scale factors and biases for the accelerometers on this platform. From the plots, it is asserted that the correlation between the two sets of signals is satisfactory for our purposes.

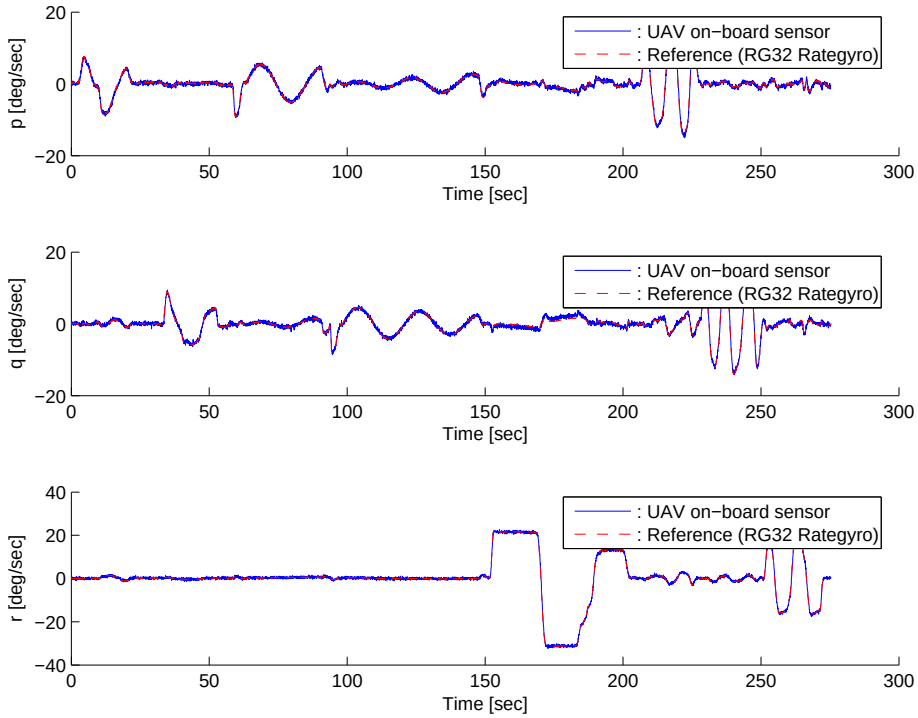


Figure 60: Angular rate calibration results.

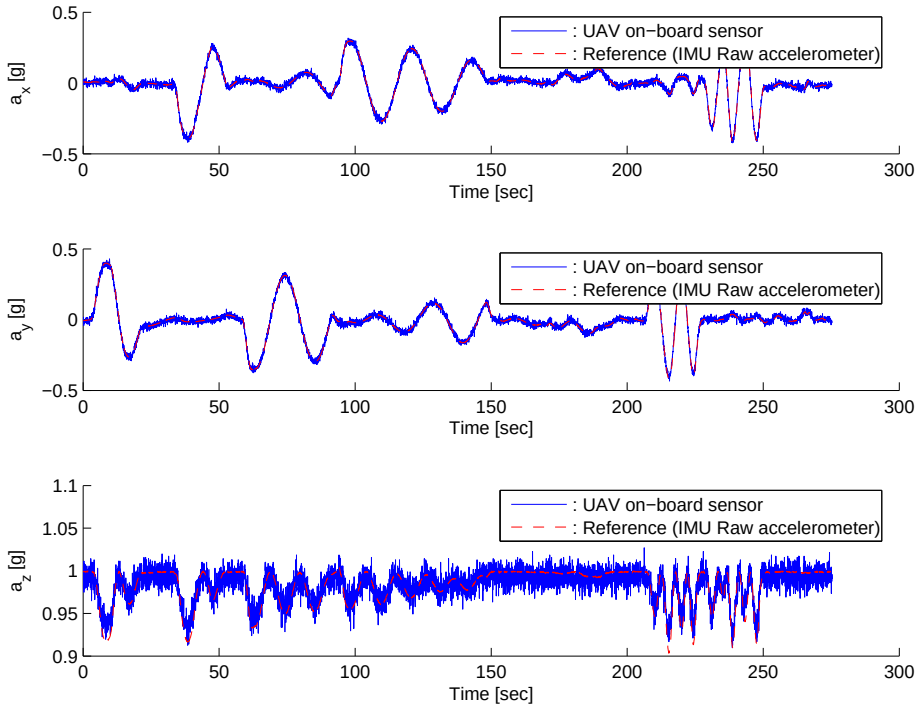


Figure 61: Accelerometer calibration results.

A.4.2 Magnetometer Calibration

The magnetometer can provide absolute orientation and it is not affected by motion constraints. On the other hand, it is susceptible to magnetic disturbances from nearby permanent magnets or ferrous materials that locally distort the Earth magnetic field. Magnetic distortion can be categorized as hard iron or soft iron effects [27]. These effects become evident as the magnetometer is rotated in the horizontal plane. By plotting the two measured signals in the body-axis frame, the hard iron distortion appears as a shift of the origin in the phase plot (X_h vs. Y_h), whereas soft iron effects appear as a distortion of a circle to an ellipse in the X_h vs. Y_h plane. Figure 62 shows the hard and soft iron distortions and the compensated magnetometer outputs.

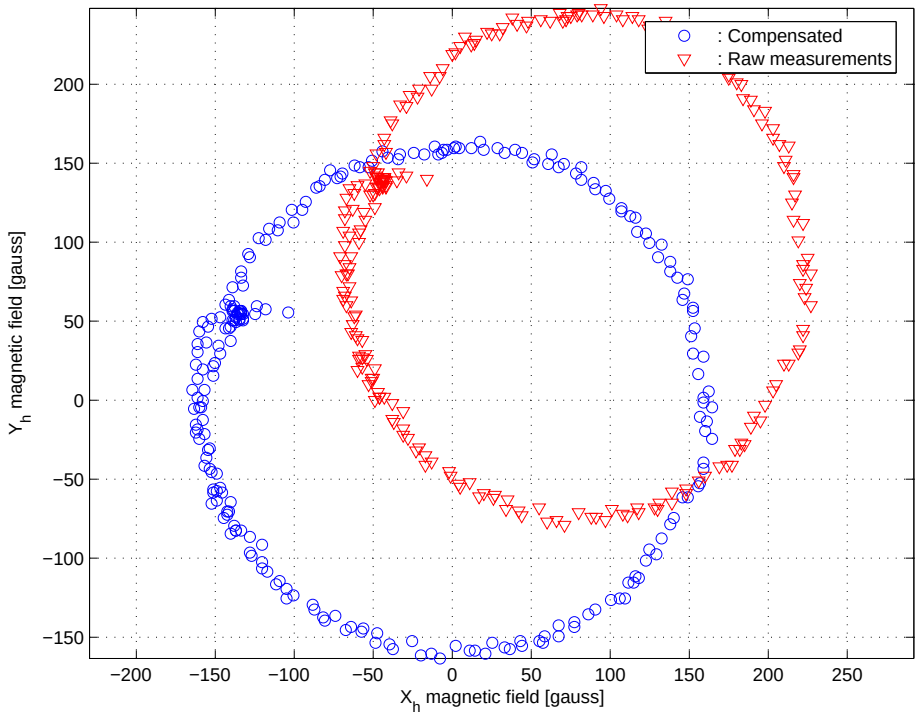


Figure 62: Effect of hard and soft iron disturbances and compensated magnetometer measurements.

Table 9: Maximum deflection angles for each control surface.

Control surface	+Range	-Range
Aileron δ_a	21.8 deg	-21 deg
Elevator δ_e	28 deg	-29.6 deg
Rudder δ_r	16 deg	-20.7 deg
Throttle δ_t	Full open (1)	Full closed (0)

A.4.3 Control Surface Deflection Calibration

One can get the actual angle of the servo motor from the corresponding voltage level of the external potentiometer linked to the DC servo motors. Therefore, the deflection angles for each control surface can be determined from each servo's position. An angle meter was used to set the actual deflection angles by a specific amount, while measuring the voltage output from the potentiometer. After several commands were applied to the servo motor, the conversion factors from potentiometer voltage level to actual deflection angle for each control surface were found. In addition, the maximum allowable deflections for each control surface were determined experimentally as well. The results are summarized in Table 9.

A.5 Summary

We have summarized the efforts undertaken to design and build a low-cost autopilot for a small UAV. The focus from the very start has been to design and assemble as much of the hardware and electronics in house as possible. This choice was opted for in order to achieve the full accessibility to the entire system, so the developed autopilot satisfies not only the stringent size and weight constraints to fit in a small UAV but also it can provide full functionality for an autonomous UAV operation.

APPENDIX B

INERTIAL ATTITUDE AND POSITION REFERENCE SYSTEM DEVELOPMENTS

B.1 Attitude estimation

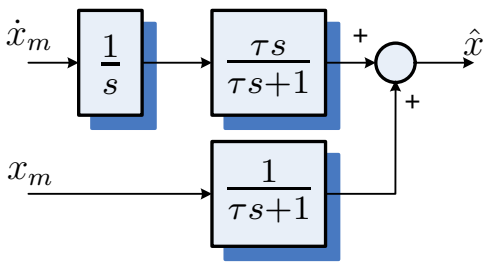
The sensors involved in a strapdown attitude and heading reference system are rate gyros, accelerometers and magnetometers. These sensors measure the three-axis angular rates, three-axis apparent acceleration (gravity minus inertial acceleration), and Earth's magnetic field with respect to the body frame. In order to obtain the best estimate of the attitude angles from the available sensors, it is imperative to blend these measurements in a seamless manner by taking into account the different signal specifications for each sensor.

B.1.1 Complementary filter

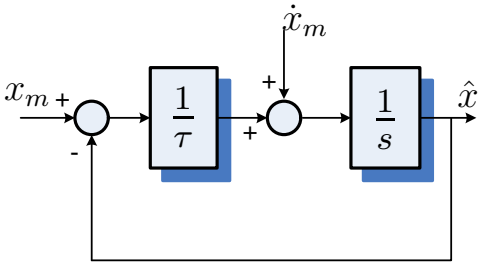
Complementary filters have been widely used to combine two independent noisy measurements of the same signal, where each measurement is corrupted by different types of spectral noise[20]. The filter provides an estimate of the true signal by employing two complementary high-pass and low-pass filters. Figure 63(a) shows the case of using a complementary filter to obtain an estimate $\hat{x}(t)$ of $x(t)$ from the two measurements $x_m(t)$ and $\dot{x}_m(t)$. Notice that $x_m(t)$ is the measurement of the signal with predominantly high-frequency noise $n_1(t)$ and $\dot{x}_m(t)$ is the measurement with low-frequency noise $n_2(t)$ as follows

$$x_m(t) = x(t) + n_1(t) \quad \text{and} \quad \dot{x}_m(t) = \dot{x}(t) + n_2(t). \tag{125}$$

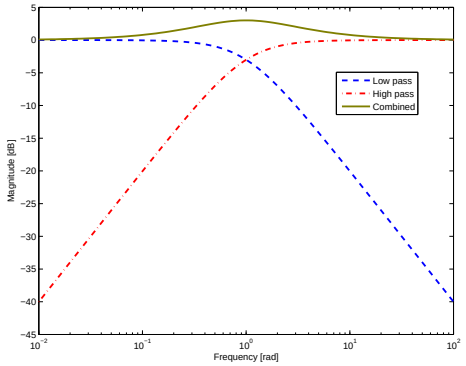
From Fig. 63(a), it is apparent that the Laplace transform of the estimate can be written as



(a) Direct complementary filter



(b) Indirect complementary filter



(c) Frequency magnitude plot for the output of a complementary filter

Figure 63: Two different schemes for the implementation of the complementary filter.

$$\hat{X}(s) = \underbrace{\frac{1}{\tau s+1}X(s) + \frac{\tau s}{\tau s+1}X(s)}_{\text{Signal terms}} + \underbrace{\frac{1}{\tau s+1}N_1(s) + \frac{\tau s}{\tau s+1}\Big(\frac{1}{s}N_2(s)\Big)}_{\text{Noise terms}} \tag{126}$$

The noise terms in both channels are effectively suppressed by the first order low- and high- pass filter with the time constant τ. The frequency response plot shown in Fig. 63(c) illustrates the contribution of each frequency channel to the output, where the cutoff frequency is chosen as 1 [rad/sec]. The time constant τ is selected according to the noise characteristics of each channel such that the estimate $\hat{x}$ is obtained by integrating $\dot{x}_m$ over frequencies $\omega \gg 1/\tau$, whereas for frequencies $\omega \ll 1/\tau$, $\hat{x}$ tracks x_m. At frequencies near $1/\tau$ the estimated output $\hat{x}$ is a combination of the two channels, which appears as a hump in Fig. 63(c). The estimate approximates the true signal faithfully over most of the frequency range.

In order to implement a complementary filter on a micro-controller, a discrete version of the high-pass and low-pass filters should be written in software by taking into consideration the sampling period of the micro-controller. The filter coefficients of the discrete filters are related to the cutoff frequency τ, which forces the user to recalculate the coefficients of both filters, if necessary. Instead, an alternative form of the complementary filter can be used as depicted in Fig. 63(b). The filter transfer function remains the same as in Eq. (126) but the feedback structure of the filter simplifies the filter implementation on a micro-controller. It also allows easy tuning for acceptable performance when low cost sensors are used. In addition, this feedback structure can be easily adapted to deal with multiple measurements. Figure 64 illustrates the case when using two low frequency channels. A tuning parameter $\alpha_k \in [0, 1]$ sets the relative weight between the signals x_{m_1} and x_{m_2}. By choosing the more reliable measurements the filter takes advantage of multiple measurements to calculate the best estimate.

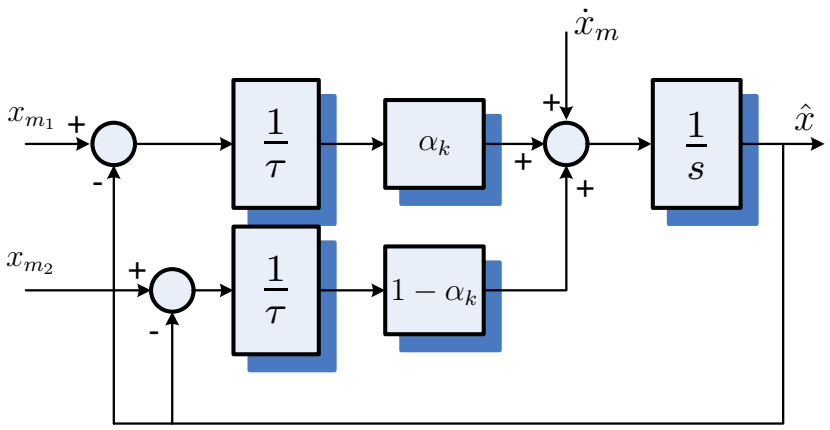


Figure 64: Multiple measurements augmentation in the indirect complementary filter.

B.1.2 Pitch and heading angle estimation

The low frequency dominant pitch angle is directly calculated from accelerometer outputs because the accelerometer is able to measure the gravity vector minus the inertial acceleration (the apparent acceleration, $\vec{g}-\vec{a}_I$) with respect to the body axes. In steady-state flight conditions, the accelerometers output mostly the gravity vector since the inertial acceleration is negligible at the steady state. Then the pitch angle is calculated from the accelerometer output $\mathbf{a} = [a_x \; a_y \; a_z]^\mathsf{T}$ as follows

$$\theta_L = -\sin^{-1}\left(\frac{a_x}{g}\right). \tag{127}$$

The low frequency dominant heading angle is determined by two different sources: the GPS sensor and the magnetometer. The GPS sensor used in this research provides an absolute heading information ψ_{GPS} at a low rate (1 Hz), the output of the three-axis magnetometer $\mathbf{m} = [m_x \; m_y \; m_z]^\mathsf{T}$ with respect to the body axes provides a heading

measurement ψ_L at a much higher rate according to the following relationship[26]

$$\psi_L = \begin{cases} \pi - \tan^{-1}(\overline{m}_y/\overline{m}_x) & \text{if } \overline{m}_x < 0, \\ 2\pi - \tan^{-1}(\overline{m}_y/\overline{m}_x) & \text{if } \overline{m}_x > 0,\ \overline{m}_y > 0, \\ -\tan^{-1}(\overline{m}_y/\overline{m}_x) & \text{if } \overline{m}_x > 0,\ \overline{m}_y < 0, \\ \pi/2 & \text{if } \overline{m}_x = 0,\ \overline{m}_y < 0, \\ 3\pi/2 & \text{if } \overline{m}_x = 0,\ \overline{m}_y > 0, \end{cases} \tag{128}$$

where $\overline{m}_x$ and $\overline{m}_y$ are the projected magnetic field components on the horizontal plane that can be calculated by transforming $\mathbf{m}$ through the rotation matrix $\mathcal{C}(\phi,\theta) \triangleq \big(\mathcal{C}_1(\phi)\mathcal{C}_2(\theta)\big)^{\mathsf{T}}$. If the pitch angle θ and the roll angle ϕ are not available, their estimates $\hat{\theta}$ and $\hat{\phi}$ can be used instead, to yield

$$\begin{aligned} \overline{m}_x &= m_x \cos\hat{\theta} + m_y \sin\hat{\phi}\sin\hat{\theta} + m_z \cos\hat{\phi}\sin\hat{\theta} \\ \overline{m}_y &= m_y \cos\hat{\phi} - m_z \sin\hat{\phi} \end{aligned} \tag{129}$$

The high frequency dominant pitch and heading angles are inferred from the attitude kinematics equations,

$$\begin{aligned} \dot{\theta} &= q\cos\hat{\phi} - r\sin\hat{\phi} \\ \dot{\psi} &= (q\sin\hat{\phi} + r\cos\hat{\phi})/\cos\hat{\theta} \end{aligned} \tag{130}$$

where, $\boldsymbol{\omega} = [p\ q\ r]^{\mathsf{T}}$ is the onboard rate gyro measurement.

Figure 65 illustrates the block diagram for the combined complementary filters for pitch and heading angle estimation. As discussed earlier, the filters can be tuned for acceptable performance via the parameters τ_θ and τ_ψ for pitch and heading angle, respectively. The relative weight for the heading angle between the magnetometer and the GPS sensor is imposed by the parameter $\alpha_\psi \in [0\ , 1]$ in order to put more emphasis on the measurement that seems to be close to the true heading. A detailed description regarding the adaptive tuning of α_ψ is given later.

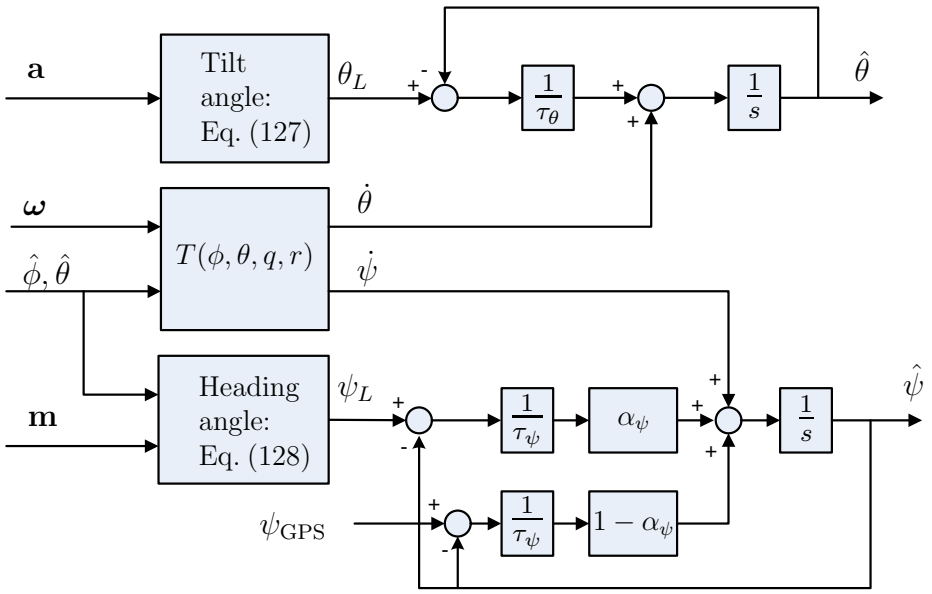


Figure 65: Entire complementary filter setup for pitch and heading angles.

B.1.3 Roll angle estimation

The roll angle can also be estimated from a complementary filter using high frequency dominant information for $\dot{\phi}$ via the attitude kinematics and low frequency dominant information from the kinematic relationship of the airplane at a banked condition. As illustrated in Fig. 66, if an airplane is in a purely banked, coordinated turn con-

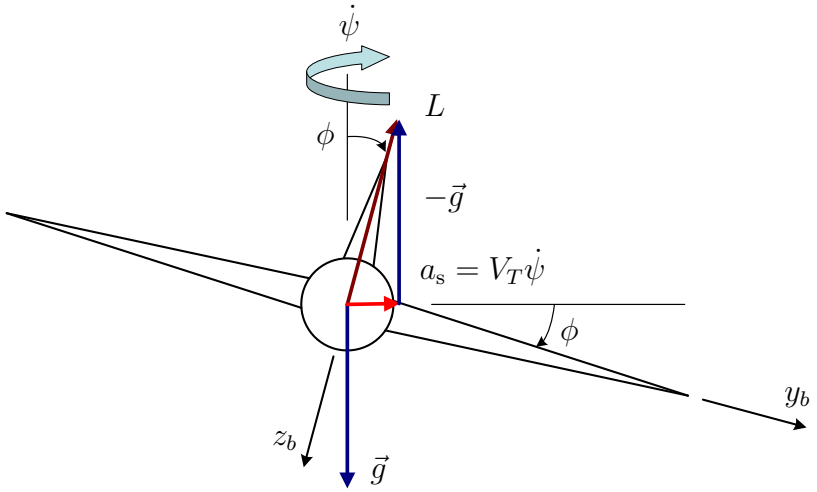


Figure 66: Kinematic relation at a truly banked turn condition.

dition (no side acceleration along body y-axis), then the roll angle is approximately

computed by the following relation assuming no wind[41],

$$\sin\phi = \frac{\dot{\psi} V_T}{g} \tag{131}$$

where V_T is the flight speed and g is the gravitational acceleration. This equation can be further approximated using $\sin\phi \approx \phi$ and $\dot{\psi} \approx r$ as

$$\phi = \frac{r V_T}{g}. \tag{132}$$

Then the low frequency dominant roll angle is approximated from Eq. (132) with the yaw rate from the yaw gyro and the flight speed from the pitot tube. In reality, the estimate from Eq. (132) tends to be biased since it utilizes the direct gyro output which is vulnerable to drift. Over a long period of time the estimate will deviate owing to this yaw rate bias[105]. To compensate for this bias, a Kalman filter was designed as follows.

In general, the fast roll dynamics of the airplane allow to use a linear approximation for the roll kinematics $\dot{\phi} = p$. The roll rate gyro measurement p_m is assumed to be corrupted by the roll rate bias p_b as well as measurement noise η_p,

$$p_m = p + p_b + \eta_p.$$

The biases for both roll and yaw gyros are modeled as random walk processes driven by Gaussian white noise processes. It follows that the equations of the filter dynamics are given by

$$\begin{aligned} \dot{\phi} &= p_m - p_b - \eta_p, \\ \dot{p}_b &= \epsilon_p, \\ \dot{r}_b &= \epsilon_r. \end{aligned} \tag{133}$$

The measurement model of the Kalman filter includes the yaw gyro output and the rate of heading angle change induced from the heading angle measurement by the

GPS sensor. The yaw rate gyro measurement r_m contains a bias r_b and measurement noise η_r

$$r_m = r + r_b + \eta_r. \tag{134}$$

Hence the yaw rate measurement is related to the roll angle, using Eq. (132), as follows,

$$r_m = \frac{g}{V_T}\phi + r_b + \eta_r. \tag{135}$$

On the other hand, because the GPS sensor provides the heading angle at one second interval, the rate of heading angle change $\dot{\psi}$ is calculated through numerical differentiation. It follows from Eq. (131) that

$$\dot{\psi}_m = \frac{g\cos\hat{\phi}}{V_T^*}\phi + \eta_{\dot{\psi}}, \tag{136}$$

where, V_T^* is the flight speed obtained from the GPS sensor, $\eta_{\dot{\psi}}$ is the measurement noise, and $\hat{\phi}$ is the current roll angle estimate. Equations (135) and (136) become the measurement model for roll angle Kalman filter. The Kalman filter is implemented in a discrete format on the micro-controller using a sampling period Δt:

- **Time update**
  - Project ahead

$$\begin{aligned} \hat{\mathbf{x}}_k^- &= \Phi_k \hat{\mathbf{x}}_{k-1} + [\ \Delta t p_m^k \quad 0 \quad 0\ ]^\mathsf{T}, \\ \mathbf{P}_k^- &= \Phi_k \mathbf{P}_{k-1} \Phi_k^\mathsf{T} + \mathbf{Q}_k, \end{aligned} \tag{137}$$

    where,

$$\Phi_k = \begin{bmatrix} 1 & -\Delta t & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad \hat{\mathbf{x}}_k = \begin{bmatrix} \hat{\phi}_k \\ \hat{p}_{b_k} \\ \hat{r}_{b_k} \end{bmatrix}.$$

- **Measurement update**

– Compute Kalman gain

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^{\star\mathsf{T}} \left(\mathbf{H}_k^\star \mathbf{P}_k^- \mathbf{H}_k^{\star\mathsf{T}} + \mathbf{R}_k^\star \right)^{-1}, \tag{138}$$

– Update estimate with measurements

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{z}_k^\star - \mathbf{H}_k^\star \hat{\mathbf{x}}_k^-), \tag{139}$$

– Compute error covariance for updated estimate

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k^\star) \mathbf{P}_k^-, \tag{140}$$

where, $\star = s,\ f$.

Note that the measurement update for GPS is done once every second at which the new $\dot{\psi}_m$ becomes available, whereas the measurement update for the yaw rate gyro occurs at a high update rate. These two measurement updates are coordinated such that each update is completed whenever the corresponding measurement becomes available: a fast update for r_{m_k} and a slow update for $\dot{\psi}_{m_k}$ as follows,

- Fast update

$$\mathbf{z}_k^f = r_{m_k}, \qquad \mathbf{H}_k^f = \begin{bmatrix} \dfrac{g}{V_T} & 0 & 1 \end{bmatrix}, \tag{141}$$

- Slow update

$$\mathbf{z}_k^s = \dot{\psi}_{m_k}, \qquad \mathbf{H}_k^s = \begin{bmatrix} \dfrac{g}{V_T} \cos \hat{\phi}_k^- & 0 & 1 \end{bmatrix}. \tag{142}$$

The process noise covariance matrix $\mathbf{Q}_k$ and measurement noise covariance matrix $\mathbf{R}_k$ are determined from the noise characteristics of each signal by assuming that the noise processes are uncorrelated to each other, as follows

$$\begin{aligned} \mathbf{Q}_k &= \mathrm{diag}\Big(E\big[\begin{array}{ccc} \bar{\eta}_p^2 & \bar{\epsilon}_p^2 & \bar{\epsilon}_r^2 \end{array} \big] \Big), \\ \mathbf{R}_k^f &= E\big[\bar{\eta}_r^2\big], \qquad \mathbf{R}_k^s = E\big[\bar{\eta}_{\dot{\psi}}^2\big], \end{aligned} \tag{143}$$

where the overbar variables represent the discrete noise sequences at the sampling period of Δt such that they have equal noise strength as the continuous noise process, and $E[\cdot]$ calculates the mean-square noise strength.

B.1.4 Dealing with GPS latency and GPS lock

The GPS receiver employed in this research has an inherent data latency, which causes the output of the GPS sensor to be delayed by a certain amount of time. The position and velocity output from the GPS sensor at the kth time step are processed internally based on the satellite range measurements at the epoch of the $(k-1)$th time step. Combining the latency due to the internal data processing along with the communication latency, the time delay of the position output is observed to be about 0.1 [sec] and the delay of the velocity output about 1.1 [sec][98]. The GPS sensor also provides the heading angle based on the internal velocity estimate. Hence, the measurement of ψ_{GPS} is also delayed by 1.1 [sec]. Unless this delay is properly compensated, substantial errors will arise during the estimation process.

Figure 67 illustrates graphically the issue of delay. Assume that the measurement is delayed by N samples. Then the measurement $\mathbf{z}_k^*$ at $t = t_k$ represents the state of the N prior sample $\mathbf{z}_k^* = \mathbf{H}_{k-N}^{\star}\mathbf{x}_{k-N}$. Several arrangements to incorporate delays into the Kalman filter framework have been suggested in the literature[8, 79]. One approach is to simply recalculate the complete time-trajectory of the filter throughout the delayed period when a delayed measurement is received. The large memory cost for storing the intermediate states over the delayed period and the corresponding increased computational cost prevent delayed measurements from being incorporated directly in a real-time estimation algorithm. Another approach to account for delayed

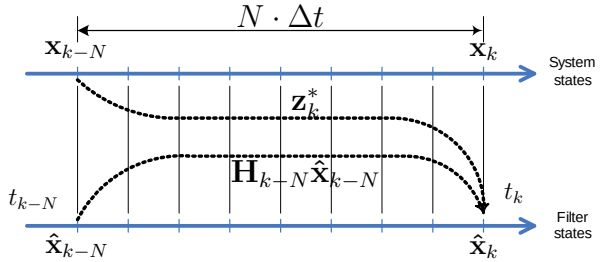


Figure 67: System with a delayed measurement due to sensor latency.

measurements is to make use of the state estimate corresponding to the delayed

measurement at $t = t_{k-N}$, i.e., $\hat{\mathbf{x}}_{k-N}$ in a buffer. When the delayed measurement $\mathbf{z}_k^*$ becomes available at the time $t = t_k$, an innovation that is calculated from the delayed measurement and the delayed state estimate in the buffer is blended with the current estimate state in the standard Kalman measurement update as follows,

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k^* - \mathbf{H}_{k-N}\hat{\mathbf{x}}_{k-N}). \tag{144}$$

Compared to the case of a wrong innovation calculated from the delayed measurement and the current state estimate $\hat{\mathbf{x}}_k^-$ (no delay compensation) as in Eq. (139), this approach forces the correct innovation to be used in the measurement update and then yields a better estimate. It is, however, a sub-optimal solution[79].

The estimation filters presented in the previous section assumed that the GPS measurement is always available. However, the GPS output is locked when a GPS outage occurs, failing to provide successive information. The situation gets even worse if the UAV performs aggressive motion such as a sharp turn at high speed. During such an outage, the GPS output is held at the previous valid output. Figure 68 demonstrates the real measured GPS data when the UAV performs a sharp turn by a remote pilot. The plot shows that at $t = 2711.2$ [sec] the GPS heading angle is held to the previous value of -90 [deg]. The value is kept until $t = 2714.2$ [sec] when a new (possibly valid) heading angle is provided by the GPS sensor. Because the attitude estimation algorithm presented earlier utilizes the heading angle measurement from the GPS sensor, an incorrect heading information from the GPS sensor due to a GPS outage would result in a wrong estimation of the heading angle in the complementary filter. Moreover, this would lead to a wrong estimation of the roll angle from the Kalman filter.

One possible remedy to this is to use intermitently the heading information from the magnetic compass for a short time period (i.e., during GPS outage). Even though any local magnetic distortion due to the existence of ferrous materials near the magnetic compass yields a small offset in the magnetic heading information, the magnetic

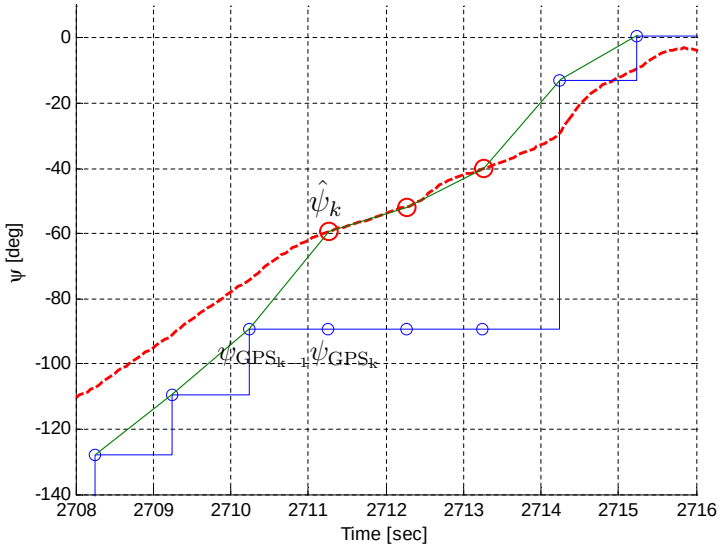


Figure 68: GPS momentary outage during an aggressive maneuver.

heading information can provide continuous heading measurement to the filtering algorithm even when a GPS outage occurs. Whenever the complementary filter detects the GPS heading angle being held constant, it first modifies the weighting parameter α_ψ to put more emphasis on the magnetic heading information. As the α_ψ approaches one, the magnetic heading information is used more exclusively in the complementary filter, while the GPS heading information is ignored. A update logic for a new weighting parameter α_ψ^* is simply given by

$$\alpha_\psi^* = \alpha_\psi + (1 - \alpha_\psi)\lambda \tag{145}$$

where, $\lambda \geq 1$ controls the rate of change of weighting parameter. Once the GPS is back to normal operation, the weighting parameter will restore the specified value for a normal operation. After the complementary filter computes the heading estimate during a GPS outage, the Kalman filter can make use of the heading estimate from the complementary filter in order to obtain the rate of change of the heading angle. Figure 68 describes the use of heading estimate from the complementary filter to compute the rate of heading angle change during a GPS outage. Notice that at

$t = 2711.2$ sec, a GPS outage occurs and the heading angle is held fixed. Then the Kalman filter switches to using the heading estimate $\hat{\psi}_k$ instead of the false heading $\psi_{\mathrm{GPS_k}}$ to calculate the rate of change of the heading angle as follows,

$$\dot{\psi}_k^* = (\hat{\psi}_k - \psi_{\mathrm{GPS}_{k-1}}). \tag{146}$$

The use of estimated heading continues until the GPS sensor yields correct heading information at $t = 2714.2$ [sec].

B.1.5 Attitude filter validation

The previous algorithm was written in C codes and implemented as an S-function in the Matlab/Simulink environment. This enables the actual C codes to be validated for any errors and tuned before used for the UAV. A complete non-linear 6-DOF Simulink model[58] is used to simulate the full dynamics of the UAV. The inertial sensor measurements are emulated to have close correlation to the real sensors used in the UAV in terms of signal specifications and noise characteristics. The gain parameters for the pitch and heading complementary filters were chosen as

$$\tau_\theta = 2, \quad \tau_\psi = 0.5, \quad \alpha_\psi = 0.4.$$

The process noise covariance matrix and the noise covariance matrix for the Kalman filter were carefully chosen in consideration of the noise characteristics of the sensors as follows

$$\begin{aligned} \mathbf{Q}_k &= \mathrm{diag}\Big(\begin{bmatrix} 0.02^2 & 0.001^2 & 0.01^2 \end{bmatrix} \Big), \\ \mathbf{R}_k^f &= 0.02^2, \qquad \mathbf{R}_k^s = 0.05^2. \end{aligned} \tag{147}$$

Two internal PID controller loops were designed for roll angle control and pitch angle control with the associated stability augmentation dampers. The filter outputs are fed back to the corresponding PID controllers and validated in the closed loop. Figure 69 shows the comparison between true values and estimated values.

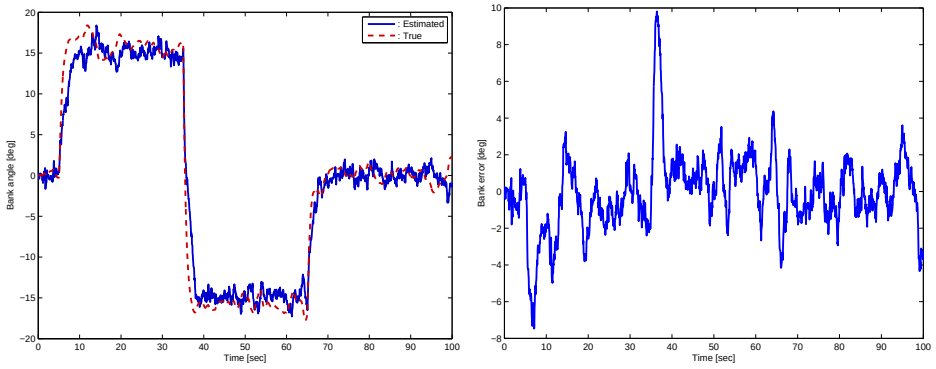
In Figs. 69(a) and 69(c) a doublet roll angle reference command was used to excite the lateral motion of the airplane, while the pitch attitude is held constant at zero. Both the complementary filter and the Kalman filter operate properly. However, a transient time lag in the estimation process is observed when the UAV changes its orientation quickly, which leads to the increased estimation error shown in the right side of Fig. 69 during the transients. Nonetheless, the filter converges to the correct angle after the UAV comes into steady state. Figure 69(b) shows the pitch angle estimation when a doublet pitch angle reference command was used to excite the longitudinal motion of the UAV, while the roll angle is controlled to zero.

B.2 Position Estimation

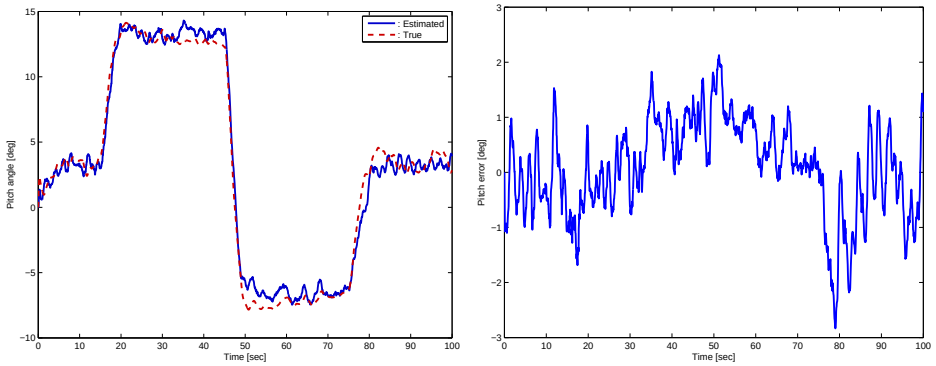
In this section a filter for estimating the absolute position of the UAV in the north-east-down (NED) inertial reference frame is developed. A typical attitude heading reference system/inertial navigation system (AHRS/INS) makes use of the accelerometer output in conjunction with the full equations of motion to propagate the inertial position and velocity from acceleration measurements using a Kalman filter. However, the dimension of this complete Kalman filter is too large to be implemented on a micro-controller and run in real-time. Instead of using the full equations of motion, the navigation equations are used to propagate the position from the flight speed measurement. The position filter is cascaded with the attitude filters so as to allow separate filter tuning and at the same time to reduce the computational cost with minimal loss of performance.

B.2.1 Filter formulation

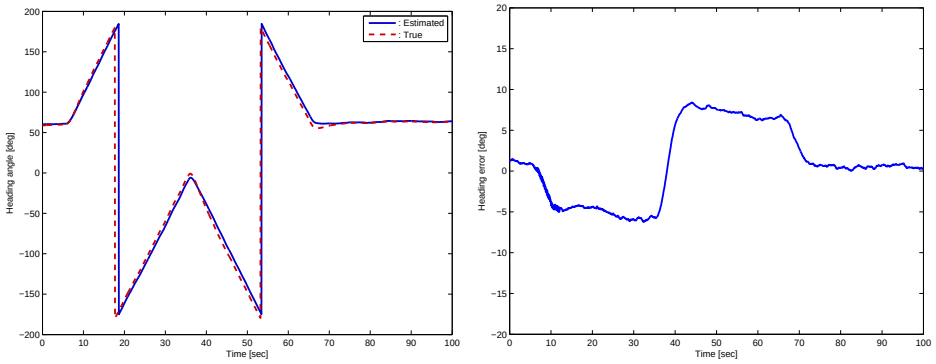
The navigation equations of a 6-dof airplane[137] are used to obtain the position filter equation. Using the rotational transform matrix from the body-axes ($\mathcal{B}$) to the



(a) Roll angle filter performance



(b) Pitch angle filter performance



(c) Yaw angle filter performance

Figure 69: Attitude estimation filters validation.

inertial frame ($\mathcal{N}$), the navigation equations are given by

$$\begin{bmatrix} \dot{p}^N \\ \dot{p}^E \\ \dot{p}^D \end{bmatrix} = \begin{bmatrix} v^N \\ v^E \\ v^D \end{bmatrix} = \left[{}^{\mathcal{B}}\mathbf{C}^{\mathcal{N}} \right]^{\mathrm{T}} \begin{bmatrix} U \\ V \\ W \end{bmatrix}. \tag{148}$$

Assume that the angle of attack and side slip angle are small, the velocity components expressed in terms of the body-axes are approximated by

$$U \approx V_T, \qquad V, W \approx 0. \tag{149}$$

The flight speed V_T, if it is measured from a pitot tube, includes the inertial speed (relative ground speed) and the wind speed. The relative ground speed is only to be integrated to propagate the inertial position. The wind speed is dependent on the flight condition, so is added to the position filter as an extra state to account for the pitot speed measurement. A random walk model for the wind speed variation is assumed, driven by a Gaussian white noise process as follows,

$$\dot{V}_w = \epsilon_w. \tag{150}$$

The speed measurement V_{T_m} from the pitot tube contains the inertial speed, the wind speed, and the measurement noise η_w,

$$V_{T_m} = V_T + V_{\mathrm{w}} + \eta_w. \tag{151}$$

The attitude angle information used in Eq. (148) is provided separately by the attitude filters. This cascaded configuration of the attitude and position filters reduces the complexity arising from the coupling of the variables. It therefore yields a simple position estimation filter with minimal order. It follows from Eq. (148), (149), and (151) , that the equations of filter dynamics are given by

$$\begin{bmatrix} \dot{p}^N \\ \dot{p}^E \\ \dot{p}^D \\ \dot{V}_w \end{bmatrix} = \begin{bmatrix} \cos\theta\cos\psi \\ \cos\theta\sin\psi \\ -\sin\theta \\ 0 \end{bmatrix} (V_{T_m} - V_w) + \begin{bmatrix} \nu_N \\ \nu_E \\ \nu_D \\ \epsilon_w \end{bmatrix}, \tag{152}$$

where, $[\nu_N \ \nu_E \ \nu_D]^\mathsf{T}$ is the process noise vector of η_w being projected onto the NED frame.

The measurement model for the position estimation filter is described as follows. The North position p_m^N and the East position measurements p_m^E are provided by the GPS sensor at an update rate of 1 Hz,

$$\begin{aligned} p_m^N &= p^N + \eta_N, \\ p_m^E &= p^E + \eta_E, \end{aligned} \tag{153}$$

where η_N and η_E are the measurement noise for the North and East directions, which are assumed to be the Gaussian.

Altitude information with good accuracy is attainable using a barometric altimeter. The one used in our UAV platform has minimum three-meter resolution and a higher update rate. The down position measurement p_m^D with a corresponding Gaussian noise is obtained by,

$$p_m^D = -h + \eta_h. \tag{154}$$

Having multiple measurements at different update rates, the discrete position Kalman filter implementation at a specified sampling period Δt on a micro-controller is given as follows.

- **Time update**
  - Project ahead

$$\begin{aligned} \hat{\mathbf{x}}_k^- &= \Phi_k \hat{\mathbf{x}}_{k-1} + V_w[\ \Delta t \sin\hat{\theta}_k \cos\hat{\psi}_k \quad \Delta t \cos\hat{\theta}_k \sin\hat{\psi}_k \quad -\Delta t \sin\hat{\theta}_k \quad 0\ ]^\mathsf{T}, \\ \mathbf{P}_k^- &= \Phi_k \mathbf{P}_{k-1} \Phi_k^\mathsf{T} + \mathbf{Q}_k, \end{aligned} \tag{155}$$

where,

$$\Phi_k = \begin{bmatrix} 1 & 0 & 0 & -\Delta t \cos\hat{\theta}_k \cos\hat{\psi}_k \\ 0 & 1 & 0 & -\Delta t \cos\hat{\theta}_k \sin\hat{\psi}_k \\ 0 & 0 & 1 & \Delta t \sin\hat{\theta}_k \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \hat{\mathbf{x}}_k = \begin{bmatrix} \hat{p}_k^N \\ \hat{p}_k^E \\ \hat{p}_k^D \\ \hat{V}_{w_k} \end{bmatrix}.$$

- **Measurement update**

  - Compute Kalman gain

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^{\star\mathsf{T}} \left(\mathbf{H}_k^\star \mathbf{P}_k^- \mathbf{H}_k^{\star\mathsf{T}} + \mathbf{R}_k^\star \right)^{-1}, \tag{156}$$

  - Update estimate with measurements

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{z}_k^\star - \mathbf{H}_k^\star \hat{\mathbf{x}}_k^-), \tag{157}$$

  - Compute error covariance for updated estimate

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k^\star) \mathbf{P}_k^-, \tag{158}$$

  where, $\star = s,\ f$.

Each update is performed whenever the corresponding measurement becomes available: a fast update and a slow update.

- Fast update

$$\mathbf{z}_k^f = -h_k, \qquad \mathbf{H}_k^f = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}, \tag{159}$$

- Slow update

$$\mathbf{z}_k^s = \begin{bmatrix} p_{m_k}^N \\ p_{m_k}^E \end{bmatrix}, \qquad \mathbf{H}_k^s = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \tag{160}$$

The process noise covariance matrix $\mathbf{Q_N}$ and measurement noise covariance matrix $\mathbf{R_N}$ are determined from the noise characteristics of each signal,

$$\begin{aligned} \mathbf{Q_N} &= \mathrm{diag}\Big(E\big[\ \bar{\nu}_N^2 \quad \bar{\nu}_E^2 \quad \bar{\nu}_h^2 \quad \bar{\epsilon}_w^2\ \big]\Big), \\ \mathbf{R}_\mathbf{N}^f &= E\big[\bar{\eta}_h^2\big], \qquad \mathbf{R}_\mathbf{N}^s = \mathrm{diag}\Big(E\big[\ \bar{\eta}_N^2 \quad \bar{\eta}_E^2\ \big]\Big), \end{aligned} \tag{161}$$

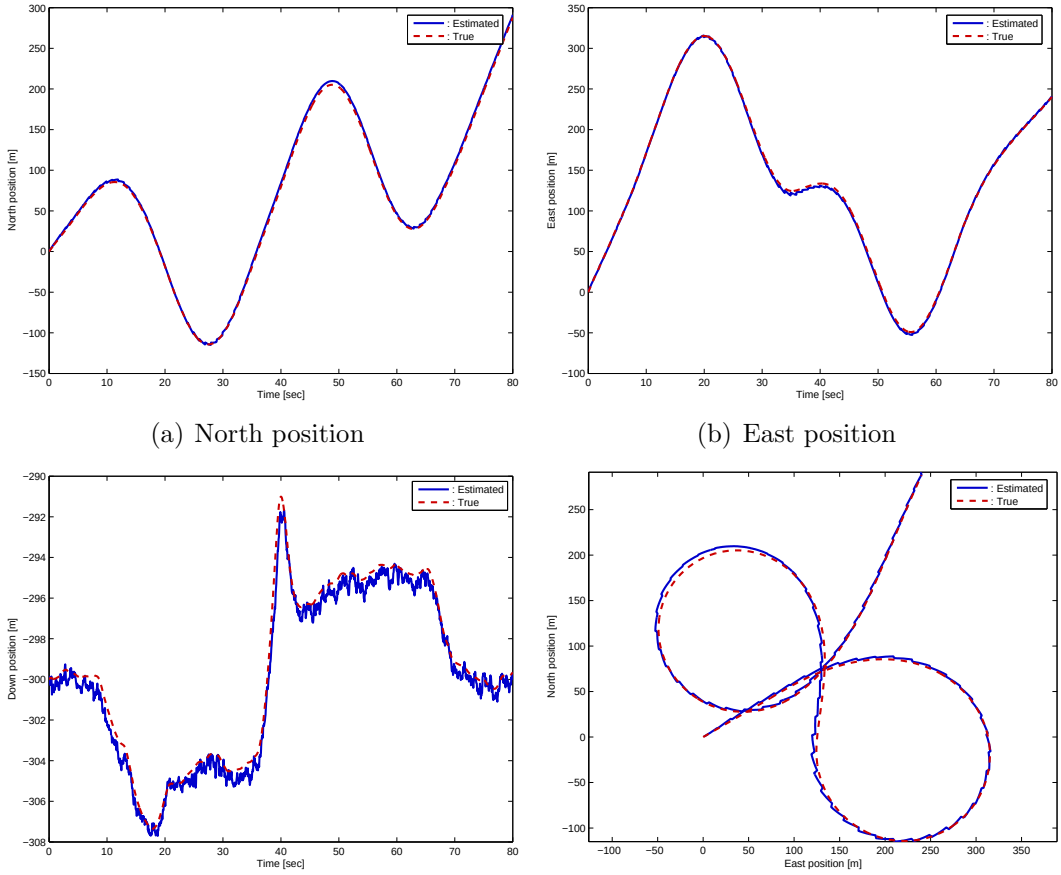
where the overbar variables represent the discrete noise sequences at a sampling period of Δt having equal noise strength as the continuous noise process, and $E[\cdot]$ calculates the mean-square noise strength.

B.2.2 Navigation filter validation

The navigation filter was written in C code as an S-function of the Matlab/Simulink® environment, as described in Section B.1.5. The process covariance matrix and the noise covariance matrix for the navigation filter were carefully chosen based on the noise characteristics of the sensors as follows

$$\begin{aligned} \mathbf{Q_N} &= \mathrm{diag}\Big(\big[\ 2^2 \quad 2^2 \quad 2^2 \quad 0.01^2\ \big]\Big), \\ \mathbf{R}_\mathbf{N}^f &= 2^2, \qquad \mathbf{R}_\mathbf{N}^s = \mathrm{diag}\Big(\big[\ 3^2 \quad 3^2\ \big]\Big). \end{aligned} \tag{162}$$

An open loop steering command for the UAV to perform an eight-shape maneuver was used, which in turn results in a doublet bank angle command as a reference to the roll PID controller. Figure 70 shows the performance of the navigation filter. Overall, the navigation filter works very well by providing a series of estimation values for the main control loop at a high rate (20 Hz) despite the low update rate of the GPS sensor (1 Hz). The North and East position estimates appear to have certain corrections periodically, which can be explained by the fact that during no GPS output update, the heading estimation from the attitude filters is used to propagate the position estimation. However, the time lag of the heading output from the attitude filters causes the navigation filter to use non-ideal heading information during propagation of the states. This results in a drifted position estimate at one second after when



(a) North position

(b) East position

(c) Down position

(d) North-East position

Figure 70: Navigation filter validation.

a new GPS measurement becomes available. The navigation Kalman filter works to update the filter states according to the new GPS measurement in order to dump out the drift and to provide the best estimate of position at all times. In addition, the altitude of the UAV is controlled by the altitude PID controller loop along with the doublet reference command, which enables the position filter to be validated with respect to the down position. The Kalman filter works properly while estimating the vertical position despite the noisy barometric altitude measurement.

B.3 Summary

A simple, yet effective attitude and position estimation algorithm has been developed for use with a low-cost UAV autopilot. Utilizing a complementary filter for estimating

the pitch and heading angles, dramatically reduces the computational burden. A minimal dimension Kalman filter estimates the roll angle. An algorithm for handling GPS lock is given and is tested to show the feasibility of real-time implementation with delayed GPS measurements. A cascaded position filter is also derived, and it is shown to be effective at incorporating the slow GPS output in order to provide a high update rate position solution. Results from both simulation and hardware validation show that the execution times of the estimation algorithms are well within the capability of the micro-controller (about 10 [msec] for the attitude filters and 3 [msec] for the navigation filter).

APPENDIX C

MODELING AND HARDWARE-IN-THE-LOOP SIMULATION

C.1 Equations of Motion of a Fixed-Wing Aircraft

The standard 6-dof equations of motion for a conventional aircraft are used for modeling and simulation of a small UAV. Flat Earth approximation[137] provides a reasonable modeling assumption when the vehicle operates over a small area. The body-axes equations are as follows:

Force equations:

$$\dot{U} = rV - qW - g\sin\theta + (X_A + X_T)/m, \tag{163a}$$

$$\dot{V} = -rU + pW + g\sin\phi\cos\theta + (Y_A + Y_T)/m, \tag{163b}$$

$$\dot{W} = qU - pV + g\cos\phi\cos\theta + (Z_A + Z_T)/m, \tag{163c}$$

Moment equations:

$$J_x\dot{p} - J_{xz}(\dot{r} + pq) + (J_z - J_y)qr = \bar{L}, \tag{164a}$$

$$J_y\dot{q} + (J_x - J_z)pr + J_{xz}(p^2 - r^2) = M, \tag{164b}$$

$$J_z\dot{r} - J_{xz}(\dot{p} - qr) + (J_y - J_x)pq = N, \tag{164c}$$

Kinematic equations:

$$\dot{\phi} = p + \tan\theta(q\sin\phi + r\cos\phi), \tag{165a}$$

$$\dot{\theta} = q\cos\phi - r\sin\phi, \tag{165b}$$

$$\dot{\psi} = (q\sin\phi + r\cos\phi)/\cos\theta, \tag{165c}$$

Navigation equations:

$$\dot{p}^N = U\mathrm{c}\theta\mathrm{c}\psi + V(-\mathrm{c}\phi\mathrm{s}\psi + \mathrm{s}\phi\mathrm{s}\theta\mathrm{c}\psi) + W(\mathrm{s}\phi\mathrm{s}\psi + \mathrm{c}\phi\mathrm{s}\theta\mathrm{c}\psi), \tag{166a}$$

$$\dot{p}^E = U\mathrm{c}\theta\mathrm{s}\psi + V(\mathrm{c}\phi\mathrm{c}\psi + \mathrm{s}\phi\mathrm{s}\theta\mathrm{s}\psi) + W(-\mathrm{s}\phi\mathrm{c}\psi + \mathrm{c}\phi\mathrm{s}\theta\mathrm{s}\psi), \tag{166b}$$

$$\dot{p}^D = -U\mathrm{s}\theta + V\mathrm{s}\phi\mathrm{c}\theta + W\mathrm{c}\phi\mathrm{c}\theta, \tag{166c}$$

where the definition of each variable is found in Ref. [137].

The aerodynamic forces and moments are obtained from the dimensionless aerodynamic coefficients at a given flight condition as follows,

$$X_A = \bar{q}SC_X, \quad Y_A = \bar{q}SC_Y, \quad Z_A = \bar{q}SC_Z, \tag{167a}$$

$$\bar{L} = \bar{q}SbC_\ell, \quad M = \bar{q}S\bar{c}C_m, \quad N = \bar{q}SbC_n. \tag{167b}$$

The analysis of the aerodynamic behavior of the aircraft, however, is better understood in the stability axes, or wind axes, system. In particular, the aerodynamic forces are easily handled in the wind axes, which is given in terms of the angle of attack α and the sideslip angle β. Hence, the force equations in wind-axes are introduced as follows[137],

$$m\dot{V}_T = T\cos(\alpha + \alpha_T)\cos\beta - D + mg_1, \tag{168a}$$

$$m\dot{\beta}V_T = -T\cos(\alpha + \alpha_T)\sin\beta - C_w + mg_2 - mV_T r_s, \tag{168b}$$

$$m\dot{\alpha}V_T\cos\beta = -T\sin(\alpha + \alpha_T) - L + mg_3 + mV_T(q\cos\beta - p_s\sin\beta), \tag{168c}$$

where, V_T is total air speed, p_s and r_s are the pitch and yaw rates projected on the stability axes, m is the mass of the vehicle, and g_1, g_2, and g_3 are the projections of the gravitational acceleration on the wind axes system. The aerodynamic forces are lift (L), drag (D), and cross wind force (C_w) while T is the thrust force exerted on the aircraft.

C.1.1 Aerodynamic coefficients for forces and moments

The aerodynamic forces and moments have a complex dependence on a number of variables resulting in complicated nonlinear correlation between each variable. Building

up the aerodynamic forces and moments as a linear sum of contributing components provides a mathematically convenient way of representing the aerodynamic forces and moments for a specified flight condition. By the same token, dimensionless aerodynamic coefficients, which are associated with the stability and control derivatives in terms of independent parameters have been widely used to represent the aerodynamic characteristic of an aircraft:

$$C_D = C_{D0} + \frac{(C_L - C_{L0})^2}{\pi e AR} + |C_{D_{\delta e}}\delta_e| + |C_{D_{\delta_a}}\delta_a| + |C_{D_{\delta r}}\delta_r|, \tag{169a}$$

$$C_Y = C_{y_\beta}\beta + C_{y_{\delta_a}}\delta_a + C_{y_{\delta_r}}\delta_r + \frac{b}{2V_T}(C_{y_p}p_s + C_{y_r}r_s), \tag{169b}$$

$$C_L = C_{L_0} + C_{L_\alpha}\alpha + C_{L_{\delta_e}}\delta_e + \frac{\bar{c}}{2V_T}(C_{L_{\dot{\alpha}}}\dot{\alpha} + C_{L_q}q), \tag{169c}$$

$$C_m = C_{m0} + C_{m_\alpha}\alpha + C_{m_{\delta_e}}\delta_e + \frac{\bar{c}}{2V_T}(C_{m_{\dot{\alpha}}}\dot{\alpha} + C_{m_q}q), \tag{169d}$$

$$C_\ell = C_{\ell_\beta}\beta + C_{\ell_{\delta_a}}\delta_a + C_{\ell_{\delta_r}}\delta_r + \frac{b}{2V_T}(C_{\ell_p}p_s + C_{\ell_r}r_s), \tag{169e}$$

$$C_n = C_{n_\beta}\beta + C_{n_{\delta_a}}\delta_a + C_{n_{\delta_r}}\delta_r + \frac{b}{2V_T}(C_{n_p}p_s + C_{n_r}r_s), \tag{169f}$$

where, e is the Oswald coefficient, AR is the main wing aspect ratio.

C.1.2 Numerical modeling of aerodynamic coefficients

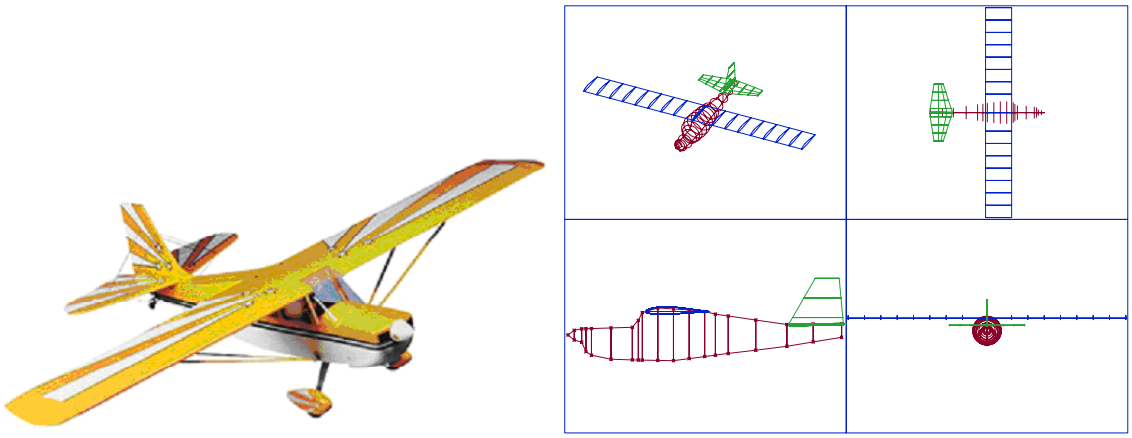
The stability and control derivatives shown in Eqs. (169) determine the aerodynamic characteristic of the aircraft, and can be estimated or identified through wind tunnel tests or via flight tests. Despite the fact that the wind tunnel test gives accurate results, a complete solution for all derivatives is not always feasible from the wind tunnel test alone. Hence, it is beneficial to obtain these derivatives from empirical data, if possible. The United States Air Force (USAF) has developed the stability and control compendium (DATCOM)[150], a large collection of information combining both classical aerodynamic analysis and experimental data. The digital DATCOM[16] is the digital version of DATCOM, a computer program which was originally written

in Fortran and ported afterwards to ANSI C. It incorporates the classical aerodynamic equations with empirical corrections from experimental data for various aircraft to compute the aerodynamic stability and control derivatives of the aircraft. The DDATCOM is useful in predicting the stability and control derivatives for preliminary aircraft designs or developing a high fidelity flight simulation[44]. The DDATCOM takes an input file containing the aircraft configuration and geometric parameters, the flight condition, the mass properties, and so on. The input configuration file of the UAV airframe used in this research is shown in Table 10.

Table 10: Digital DATCOM input configuration file of the 1/5 scale Decathlon.

Properties	value	units
Mass (m)	5.6132	[kg]
Aerodynamic reference area (S)	0.6558	[m^2]
Longitudinal reference length ($\bar{c}$)	0.3215	[m]
Lateral reference length (b)	2.04	[m]
Free stream airspeed (V_T)	20	[m/sec]
Flight path angle (γ)	0	[deg]
Altitude (h)	300	[m]

The geometric configuration of the UAV airframe was accurately measured and entered in the DDATCOM input file. This included the total weight, the shape of the wing/tail airfoil section, a cylindrical modeling of the fuselage section, the center of gravity location, the placement of wing/tails, and etc. Figure 71(a) shows the UAV airframe, a commercial Decathlon R/C model, and Fig. 71(b) shows the detailed CAD model of the airframe. With the geometric data supplied, the DDATCOM computes the static stability derivatives, the dynamic stability derivatives, and control derivatives for both the aileron and the elevator. The control derivatives associated with the rudder and the stability derivatives associated with the yawing moment are not computed from the DDATCOM. These derivatives can be estimated using the method proposed in Ref. [132]. The stability and control derivatives calculated from



(a) (b)

Figure 71: Geometric modeling of the 1/5 scale Decathlon used as an input to the DDATCOM program.

the DDATCOM are summarized in Table 11.

Table 11: Static and dynamic stability derivatives and control derivatives estimated from the geometry of the 1/5 scale Decathlon.

C_{D_0}	C_{L_0}	C_{L_α}	$C_{L_{\dot{\alpha}}}$	C_{L_q}	C_{y_β}	C_{y_p}	C_{y_r}
0.028	0.062	5.195	1.22	4.589	-0.2083	0.0057	0.0645*
C_{m_0}	C_{m_α}	$C_{m_{\dot{\alpha}}}$	C_{m_q}	C_{ℓ_β}	C_{ℓ_p}	C_{ℓ_r}	C_{n_β}
0.0598	-0.9317	-2.897	-5.263	-0.0377	-0.4625	0.0288	0.0116
C_{n_p}	C_{n_r}	$C_{D_{\delta_e}}$	$C_{D_{\delta_a}}$	$C_{D_{\delta_r}}$	$C_{L_{\delta_e}}$	$C_{y_{\delta_a}}$	$C_{y_{\delta_r}}$
-0.0076	-0.0276	0.0418	0.0†	0.0†	0.2167	0.0†	0.1096*
$C_{m_{\delta_e}}$	$C_{\ell_{\delta_a}}$	$C_{\ell_{\delta_r}}$	$C_{n_{\delta_r}}$	$C_{n_{\delta_a}}$			
-0.8551	-0.2559	0.0085†	0.0035	-0.0216†			

† : approximately zero assumed, * : estimation from the Reference [132].

C.1.3 Mass properties modeling

The mass properties can be properly estimated from a computational analysis with a help of CAD program. In contrast to the a CAD modeling, the mass properties have been identified experimentally based on the actual platform. The total weight of the UAV was measured using a precise scale, and the center of gravity location was carefully determined in the body axes with respect to the base point at the center of propeller. The mass moment of inertia about each principal axis were identified

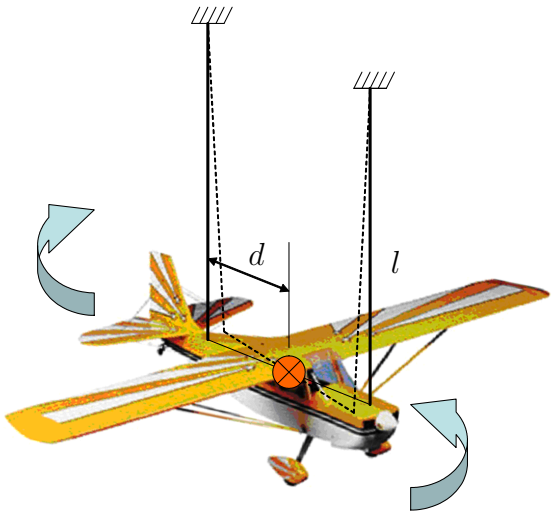


Figure 72: Torsional pendulum experimental setup for identifying the moment of inertia.

experimentally using the formula[53]

$$J_\star = \frac{gT_m^2 d^2 m}{4\pi^2 l}, \quad \star = x, y, z \tag{170}$$

where, d is the distance from the center of gravity to the point at which the suspension wire is vertically connected, T_m is the period of a small perturbed rotational motion captured by the rate gyros, and l is the length of the wire from the ceiling. Figure 72 illustrates the detail torsional pendulum experimental setup. After careful alignment of the aircraft to the horizontal plane and equal distance for each anchor point to the center of gravity, the moments of inertia were identified in the direction of x-, y-, z- of the body axes, as J_x, J_y, and J_z, respectively. Table 12 summarizes the final mass properties from experiments.

Table 12: Mass properties identified from experiments

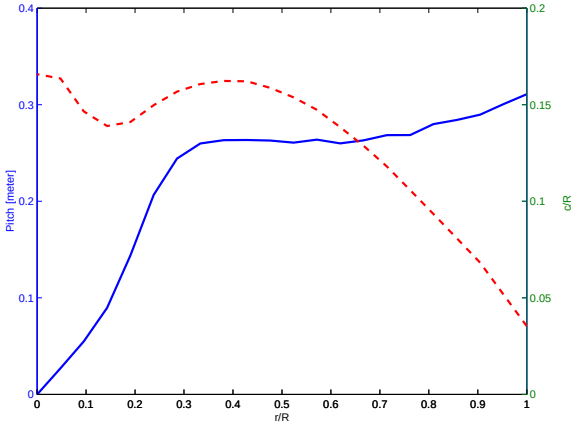
Properties	symbols	values
Mass	m	5.6132 [kg]
Center of mass location in body axes	$[r_x \;\; r_y \;\; r_z]$	$[-0.47 \;\; 0 \;\; -0.012]$ [m]
Moment of inertia	$[J_x \;\; J_y \;\; J_z]$	$[0.4497 \;\; 0.5111 \;\; 0.8470]$ [kg $\cdot$ m^2]

C.1.4 Engine and propeller modeling

The UAV is powered by a two-stroke internal combustion engine with a fixed pitch propeller. Since the thrust force is always acting on the airplane to overcome the drag even during steady level flight, comprehensive understanding on the thrust producing mechanism for the propeller-engine enables a faithful mathematical model for the simulation. The model engine is an O.S. FX 91 that delivers the maximum 2.8 horse power (HP) at a rotational speed of 16000 [rpm] with a propeller of 13 inches in diameter and 8 inches in propeller pitch distance. As a complete modeling of the engine with the propeller is beyond our scope, a simplified modeling of the thrust force exerted on the aircraft is given as follows. Assume that the engine is overpowered providing any required power to the propeller. Hence, the thrust is solely dependent on the propeller aerodynamics.

The aerodynamics of the propeller can be understood by a combination of the lift and drag force along its cross-sectional plane. The analysis of the aerodynamic forces along the propeller is based on the geometry of the propeller. The geometry for an APC model propeller of 13 inches in diameter by 8 inches in pitch distance is shown in Fig. 73. The static thrust can be computed from a program, JavaProp[92], which is written based on the blade element theory[78, 2]. With the geometry of the propeller is specified a priori, it calculates the produced thrust in terms of the rotating speed of the propeller. On the other hand, an experiment was conducted to identify the actual static thrust at different throttle command $\delta_t \in [0,\ 1]$. Figure 74 illustrates a simple experimental setup using a spring scale to directly measure the static thrust. The static thrust was approximately gauged at the steady state with a RPM reading at the given throttle command. Figure 75 shows the measured RPM v/s input throttle command. While the solid line represents actual measurements, the dashed line represents the best fitting curve that is obtained from a polynomial

(a) Horizonal and vertical view.



(b) Geometry of an APC propeller (chord length and pitch distance).

Figure 73: APC model airplane propeller 13″ by 8″.

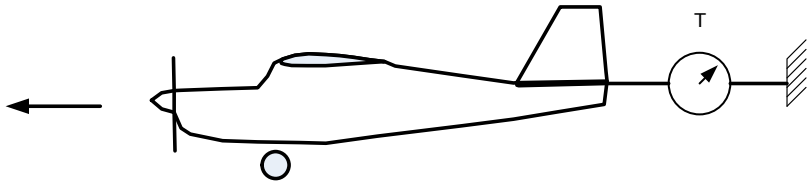


Figure 74: Experimental setup for identifying static thrust force.

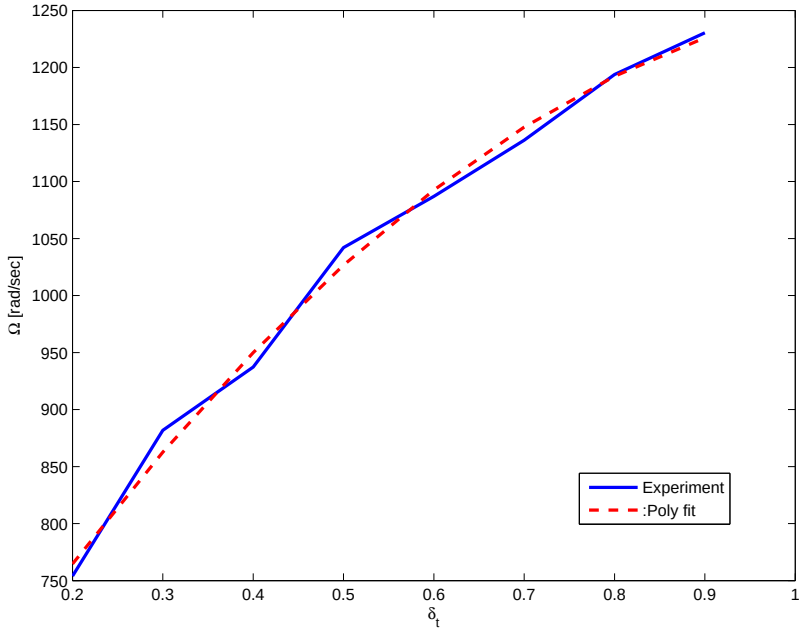


Figure 75: Rotating speed of propeller v/s the throttle command.

fit in least-square sense as follows,

$$\Omega = -534.8\delta_t^2 \; + \; 1247.5\delta_t \; + \; 536.5 \quad [\text{rad/sec}]. \tag{171}$$

Figure 76 compares the actual static thrust measured with the calculated thrust from the JavaProp. Figure 76 shows that the estimated thrust has similar trend as such that the thrust increases quadratically as the rotational speed increases. The shortage of thrust by the experiment can be attributed to the fact that the downstream after the propeller is hindered by the fuselage cross section, which affects to reduce the mass flow rate thus less thrust generated. By a simple correction, the numerical thrust can be scaled to the actual thrust by the following relation, which was also plotted by a dashed line in Fig. 76.

$$T_{\text{ex}} = 1.2407 \cdot T_{\text{th}} - 8.4742 \quad [\text{N}]. \tag{172}$$

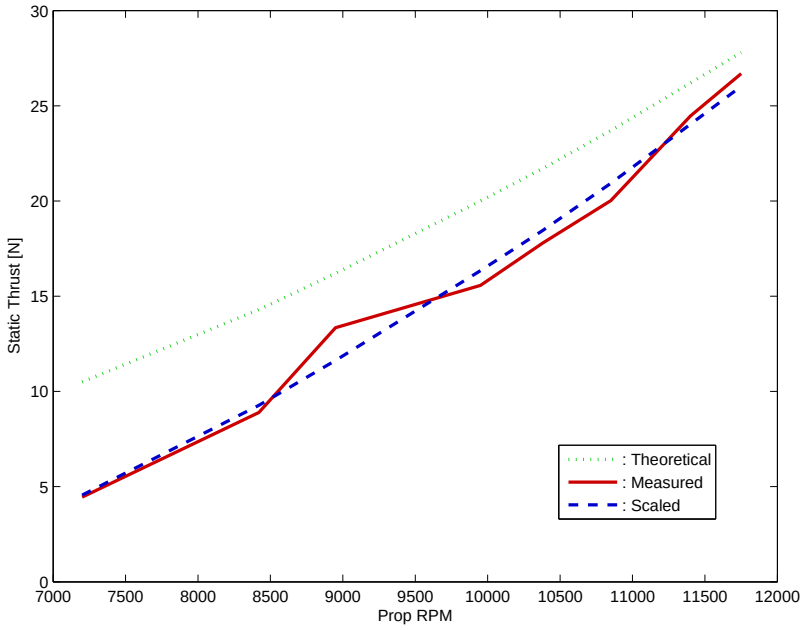


Figure 76: Static thrust comparison for a model propeller 13″ by 8″

The dynamic thrust, on the other hand, is quite complicated to be estimated directly from the propeller geometry. In general, the thrust force during flights depends not only on the rotational speed of the propeller but also on the forward flight speed (incoming wind speed). The forward flight speed reduces an effective angle of attack to the airfoil cross section, thus eventually reducing generated thrust force. This aspect will continue until a certain combination of RPM and forward flight speed at which no thrust is generated and changes its sign for an opposite direction (wind mill effect). In order to take into account this aspect, the advance ratio (J) should be introduced to incorporate the effect of forward flight speed as follows,

$$J = \frac{\pi V_T}{\Omega R_p}, \tag{173}$$

where, R_p is the propeller radius in meters and Ω the rotational speed in [rad/sec].

From the JavaProp, the thrust coefficient (c_T) was obtained in terms of the advance ratio (J) as shown in Fig. 77. This dimensionless coefficient is related to the

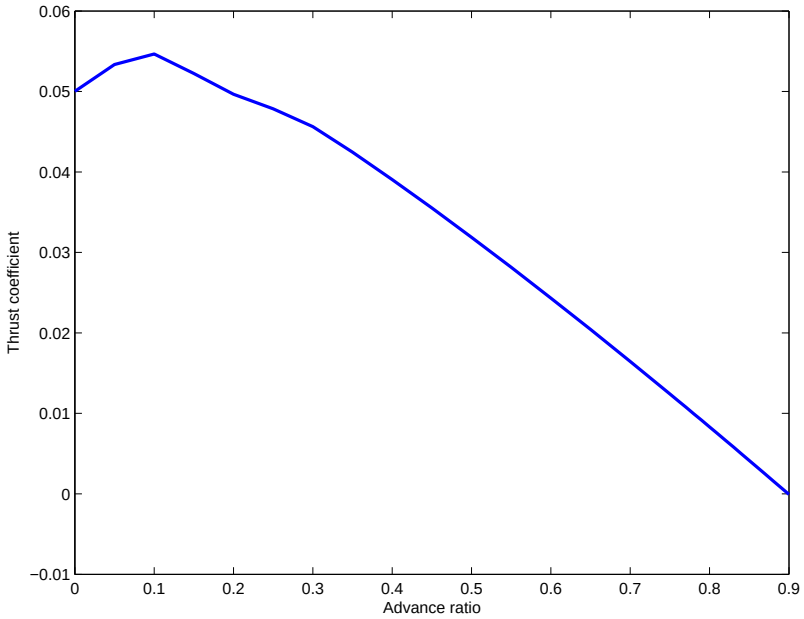


Figure 77: Thrust coefficient c_T v/s advance ratio for the selected propeller.

actual thrust by the following relation,

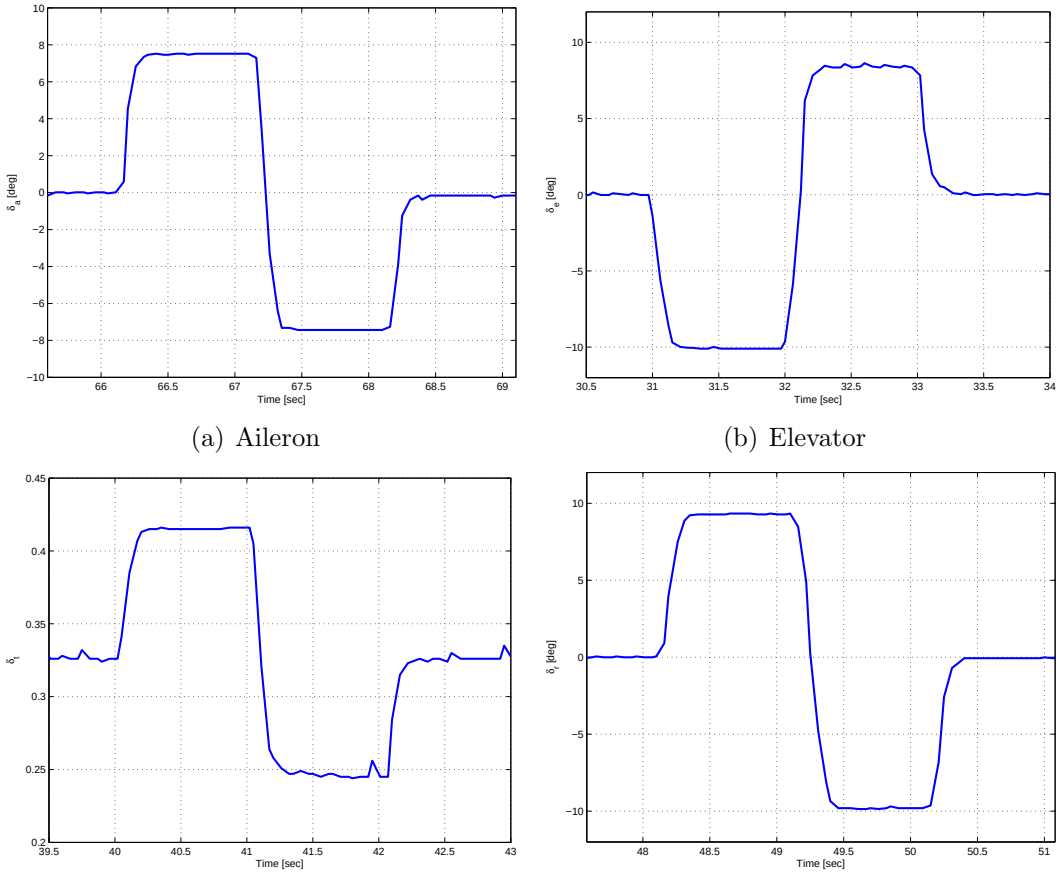
$$T_{\text{th}} = \frac{4}{\pi^2}\rho\Omega^2 R_p^4 c_T \tag{174}$$

where, ρ is the air density. Hence, the actual thrust can be estimated from Eqs. (172) and (174) using an approximation of c_T at a given advance ratio value that is computed from the flight speed and the rotational speed of propeller at each instant.

In addition, the dynamic characteristic of thrust generation is modeled by a transfer function from throttle command (δ_t) to the thrust output. The transfer function is experimentally identified by a first order low pass filter with the time constant 0.3333 [sec], which lumps together various effects such as servo motor dynamics, engine dynamics, and propeller aerodynamics.

C.1.5 Actuator modeling

The control surfaces of the UAV are actuated by four identical R/C servo motors. Each servo arm is coupled to the corresponding control surface via mechanical links.



(a) Aileron (b) Elevator

(c) Throttle (d) Rudder

Figure 78: Various doublet responses for dynamic modeling of the control surfaces.

Hence, in order to obtain the dynamic characteristics of the control surface deflection, several doublet responses were recorded for each control surface. Figure 78 shows these responses. Deflection angles of the control surfaces are measured by the potentiometers linked to the servo motor. It appears that an R/C servo motor nearly performs a dead beat control action. Thus, a first-order low pass filter combined with both slew rate limit and control saturation results in a good actuator model. Table 13 summarizes the results of dynamic modeling of the control surfaces.

C.2 Estimation of Aerodynamic Angles

The angle of attack and the sideslip angle are significant states describing the aerodynamics of the airplane. Specifically, these angles are required for identification of

Table 13: Results of dynamic modeling of each control surface identified by experiments.

Control surface	+Range	-Range	Slew rate
Aileron δ_a	21.8 deg	-21 deg	106.5 deg/sec
Elevator δ_e	28 deg	-29.6 deg	129.4 deg/sec
Rudder δ_r	16 deg	-20.7 deg	141.2 deg/sec
Throttle δ_t	Full open (1)	Full closed (0)	2.5 /sec

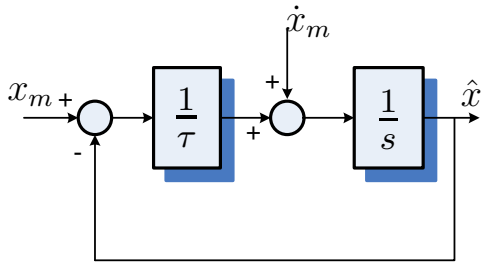


Figure 79: A complementary filter in the feedback form.

aerodynamic force and moment coefficients. For exact measurements, it is necessary to install appropriate air data sensors on-board the aircraft. However, it is sometimes difficult to install such apparatuses on a small UAV. Another approach, used here, is to determine the aerodynamic angles using inertial sensor measurements. In this section, the aerodynamic angles are estimated from raw inertial acceleration measurements using an approach similar to the one presented in Refs. [31, 51].

C.2.1 Filter formulation

Complementary filters have been widely used for combining two independent noisy measurements of a same signal, where each measurement is corrupted by different types of spectral noise[20]. Figure 79 shows a complementary filter that uses two measurements $x_m(t)$ and $\dot{x}_m(t)$ to obtain an estimate $\hat{x}(t)$ of $x(t)$. The time constant τ is selected according to the noise characteristics of each channel such that the estimate $\hat{x}$ is contributed by integration of $\dot{x}_m$ over frequencies $\omega \gg 1/\tau$, whereas for frequencies $\omega \ll 1/\tau$, $\hat{x}$ tracks x_m.

In order to employ a complementary filter in the estimation of the aerodynamic

angles, one needs to find the aerodynamic angles and their derivatives computed from the inertial measurements. Using a small angle approximation for α and β during a steady state flight condition, the body z-axis accelerometer output a_m^z measures approximately the thrust force and the aerodynamic force in the wind-axes as follows

$$ma_m^z \cong F_T \sin(\alpha + \alpha_T) + L. \tag{175}$$

It follows from Eq. (168c) that the time derivative of the angle of attack is obtained by

$$\dot{\alpha}_m = \frac{g - a_m^z}{V_T} + q. \tag{176}$$

The thrust force components in Eq. (175) can be further neglected from the assumption of small angle of attack and the thrust vector aligned to the body x-axis. Knowing that the lift force L is represented by the dimensionless coefficient of Eq (169c), one can obtain the angle of attack from the accelerometer output and the aircraft states using the following relation

$$\alpha_m = \frac{1}{C_{L_\alpha}}\Big(\frac{m}{\bar{q}S}a_m^z - \frac{\bar{c}C_{L_{\dot{\alpha}}}}{2V_T^2}(g - a_m^z) - C_{L_0} - C_{L_{\delta_e}}\delta_e - \frac{\bar{c}q}{2V_T}(C_{L_{\dot{\alpha}}} + C_{L_q})\Big). \tag{177}$$

It should be noted that a priori knowledge of the stability and control derivatives used in Eq. (177) is important to get a correct estimate of the angle of attack. These parameters can be supplied by the method discussed in Section C.1.1.

For the sideslip angle estimation, the body y-axis accelerometer output a_m^y measures approximately the thrust force and the aerodynamic force as follows

$$ma_m^y \cong F_T \cos(\alpha + \alpha_T) \sin\beta + C_w, \tag{178}$$

where C_w is the cross-wind force component. With a small angle approximation of α and β during a steady-state flight condition at small θ, it follows that $g_2 \approx g\sin\phi$ and $r_s \approx r$ in Eq. (168b). Hence the time derivative of the sideslip angle yields

$$\dot{\beta}_m = \frac{1}{V_T}\big(g\sin\phi - a_m^y\big) - r. \tag{179}$$

The thrust force components in Eq. (178) can also be neglected along the same reason discussed above. Assuming that the cross-wind force component C_w in Eq. (168b) is related to the body force component Y_A by $C_w \cong -Y_A$, it follows from Eq. (169b) that the sideslip angle is obtained from the accelerometer output and the aircraft states using the following relation

$$\beta_m = -\frac{1}{C_{y_\beta}}\Big(\frac{m}{\bar{q}S}a_m^y + C_{y_{\delta_r}}\delta_r + \frac{b}{2V_T}\big(C_{y_p}p + C_{y_r}r\big)\Big). \tag{180}$$

The estimates of the angle of attack and the sideslip angle are then computed from Eqs. (176),(177),(179) and (180) using the complementary filters, as illustrated in Fig. 79.

C.3 Identification for Aerodynamic Force/Moment Coefficients

In this section, the stability and control derivatives of the UAV are identified from the actual flight test data. In contrast to the discussion in Section C.1.1, the aerodynamic coefficients with respect to the body-axes system will be dealt with in the following formulation due to the fact that measurements are obtained via body-fixed inertial sensors. As discussed already, these coefficients are assumed to be linear in terms of each contributing component, so a linear parameterization is adopted as the identification model structure.

C.3.1 A linear parameterization

The measurements for model identification are provided by the on-board sensor suite. Among these sensors, accelerometers and rate gyros capture the dynamic behavior of the UAV, which yields the aerodynamic force coefficients expressed in the body axes as follows,

$$C_X = \frac{m}{\bar{q}S}a_m^x - \frac{T}{\bar{q}S}, \quad C_Y = \frac{m}{\bar{q}S}a_m^y, \quad C_Z = \frac{m}{\bar{q}S}a_m^z. \tag{181}$$

where the thrust force was assumed to be only exerted along the body x-axis and is modeled as a function of the rotating speed of the propeller and the forward flight speed as discussed in Section C.1.4, as follws

$$T = f(\rho,\ \Omega,\ V_T). \tag{182}$$

The aerodynamic moment coefficients are obtained as follows

$$C_\ell = \frac{\bar{L}}{\bar{q}Sb}, \quad C_m = \frac{M}{\bar{q}S\bar{c}}, \quad C_n = \frac{N}{\bar{q}Sb}, \tag{183}$$

where $\bar{L}$, M and N are calculated from Eqs. (164a), (164b) and (164c) using the body rate measurements $\boldsymbol{\omega} = [p\ q\ r]^\mathsf{T}$ and the corresponding numerical differentiation $\dot{\boldsymbol{\omega}} = [\dot{p}\ \dot{q}\ \dot{r}]^\mathsf{T}$.

A linear regression model structure was opted for by taking into account the decoupled longitudinal and lateral dynamics of the aircraft. Hence, decoupled longitudinal and lateral derivatives are associated with each aerodynamic coefficient as follows,

$$C_X = C_{x_0} + C_{x_\alpha}\alpha + C_{x_q}\frac{\bar{c}}{2V_T}q + C_{x_{\delta_e}}\delta_e \tag{184a}$$

$$C_Y = C_{y_0} + C_{y_\beta}\beta + C_{y_p}\frac{b}{2V_T}p + C_{y_r}\frac{b}{2V_T}r + C_{y_{\delta_a}}\delta_a + C_{y_{\delta_r}}\delta_r \tag{184b}$$

$$C_Z = C_{z_0} + C_{z_\alpha}\alpha + C_{z_q}\frac{\bar{c}}{2V_T}q + C_{z_{\delta_e}}\delta_e \tag{184c}$$

$$C_\ell = C_{\ell_0} + C_{\ell_\beta}\beta + C_{\ell_p}\frac{b}{2V_T}p + C_{\ell_r}\frac{b}{2V_T}r + C_{\ell_{\delta_a}}\delta_a + C_{\ell_{\delta_r}}\delta_r \tag{184d}$$

$$C_m = C_{m_0} + C_{m_\alpha}\alpha + C_{m_q}\frac{\bar{c}}{2V_T}q + C_{m_{\delta_e}}\delta_e \tag{184e}$$

$$C_n = C_{n_0} + C_{n_\beta}\beta + C_{n_p}\frac{b}{2V_T}p + C_{n_r}\frac{b}{2V_T}r + C_{n_{\delta_a}}\delta_a + C_{n_{\delta_r}}\delta_r \tag{184f}$$

The stability and control derivatives are unknown parameters to be identified, so the model structure in Eqs. 184 yields the following linear parameterization for each aerodynamic coefficient,

$$z = \boldsymbol{\theta}^\mathsf{T}\mathbf{x}, \tag{185}$$

where, z is a measurement scalar and can be obtained from Eq. (181) or (183) for each aerodynamic coefficient, $\boldsymbol{\theta}$ is a unknown parameter vector, and $\mathbf{x}$ is a regressor

matrix of which entries are comprised of the measurements from sensors along with the right-hand-side of Eqs. (184).

The problem of batch identification is to find $\boldsymbol{\theta}$ that satisfies Eq. (185). A best fit is obtained using a batch process utilizing a set of measurements $z(k)$ and $\mathbf{x}(k)^\mathsf{T}$ as follows

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}(1)^\mathsf{T} \\ \mathbf{x}(2)^\mathsf{T} \\ \vdots \\ \mathbf{x}(n)^\mathsf{T} \end{bmatrix}, \quad \mathbf{z} = \begin{bmatrix} z(1) \\ z(2) \\ \vdots \\ z(n) \end{bmatrix}. \tag{186}$$

The best fit is obtained from the least-square optimization problem,

$$\min_{\boldsymbol{\theta}} \|\mathbf{z} - \mathbf{X}\boldsymbol{\theta}\|^2, \tag{187}$$

whose solution is

$$\hat{\boldsymbol{\theta}} = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}\mathbf{z}. \tag{188}$$

C.3.2 Identification from flight test data

Flight test data were collected during maneuvers initiated from a trim condition of steady and straight level flight by exciting each control surface. Doublet-like open-loop commands to each control surface were issued by a remote pilot for each case. Upon the assumption of decoupled longitudinal and lateral dynamics, the aileron or rudder doublets were executed for identifying the lateral coefficients (C_Y, C_ℓ, and C_n) whereas the elevator doublet commands were executed for identifying the longitudinal coefficients (C_m, C_X, and C_Z). The sensor measurements were sampled at 20 [Hz] and were processed a posteriori to remove measurement noises or biases. For estimating the aerodynamic angles and attitude angles for building the regressor matrices, the estimation filters which were discussed in Section C.2 and Appendix B were utilized for off-line calculation of those variables.

Two typical sets of measured (or estimated) states for longitudinal and lateral

identification are shown in Figs. 80 and 81. The initial regression models given in Eq. (184) were first attempted for each identification, however, after several trials it was realized that slight modifications on the regression model structures in Eq. (184) result in better identification results. The modified regression model is then rearranged as follows,

$$C_X = C_{x_0} + C_{x_q}\frac{\bar{c}}{2V_T}q + C_{x_{\delta_e}}\delta_e, \tag{189a}$$

$$C_Y = C_{y_0} + C_{y_\beta}\beta + C_{y_p}\frac{b}{2V_T}p + C_{y_r}\frac{b}{2V_T}r, \tag{189b}$$

$$C_Z = C_{z_0} + C_{z_\alpha}\alpha + C_{z_{\dot{\alpha}}}\frac{\bar{c}}{2V_T}\dot{\alpha} + C_{z_q}\frac{\bar{c}}{2V_T}q, \tag{189c}$$

$$C_\ell = C_{\ell_0} + C_{\ell_\beta}\beta + C_{\ell_p}\frac{b}{2V_T}p + C_{\ell_{\delta_a}}\delta_a, \tag{189d}$$

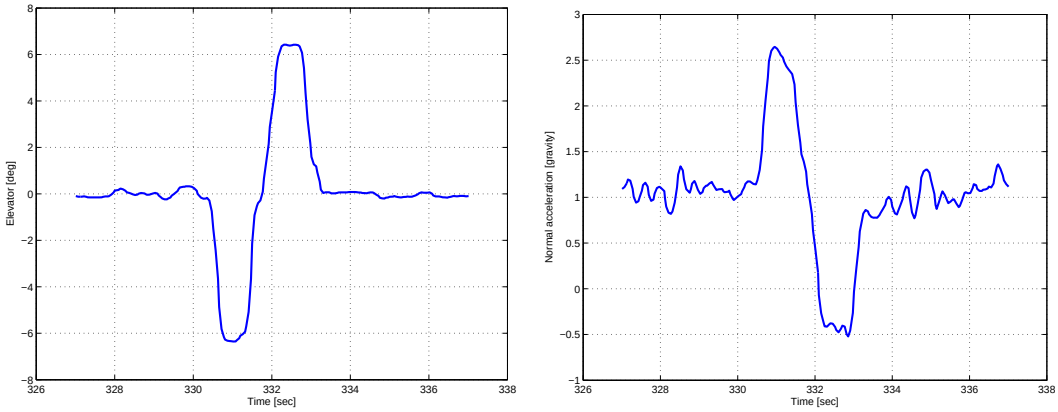
$$C_m = C_{m_0} + C_{m_\alpha}\alpha + C_{m_{\dot{\alpha}}}\frac{\bar{c}}{2V_T}\dot{\alpha} + C_{m_q}\frac{\bar{c}}{2V_T}q + C_{m_{\delta_e}}\delta_e, \tag{189e}$$

$$C_n = C_{n_0} + C_{n_\beta}\beta + C_{n_p}\frac{b}{2V_T}p + C_{n_r}\frac{b}{2V_T}r + C_{n_{\delta_a}}\delta_a + C_{n_{\delta_r}}\delta_r. \tag{189f}$$

Several data sets have been selected for each identification process to assure the correctness of the results. Subsequently, the identified parameters are found to be consistent in each case within a reasonable bound of the estimation error. The identified parameter estimates for longitudinal coefficients are shown in Tables 14-16. Note that the error value given inside the brackets represents the 1-σ standard deviation of the estimation error which provides the confidence interval of the estimate. The longitudinal coefficients result mostly in good agreement as the numerical models were validated against experiment data taken from different maneuvers shown in Figs. 82-84. Tables 17-19 summarize the identified parameter estimates for lateral coefficients.

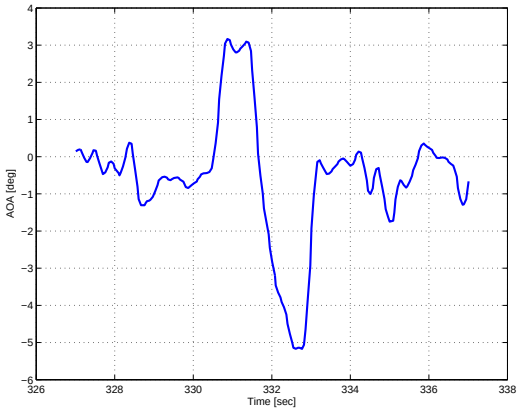
C.4 Hardware in the Loop Simulation Development

This section describes the details of developing a realistic simulation environment based on Matlab/ Simulink®. A complete 6-DOF nonlinear aircraft model with a

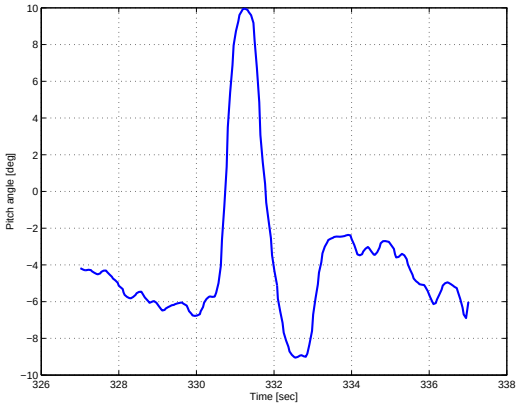


(a) Elevator command

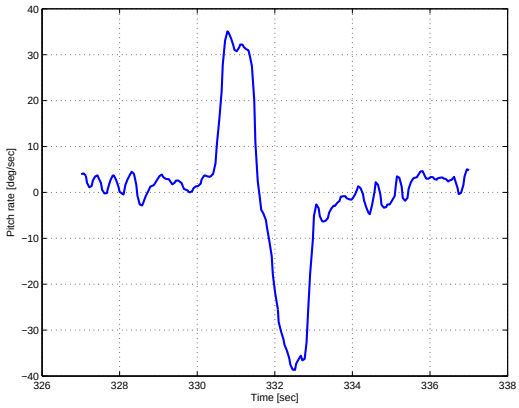
(b) Normal acceleration



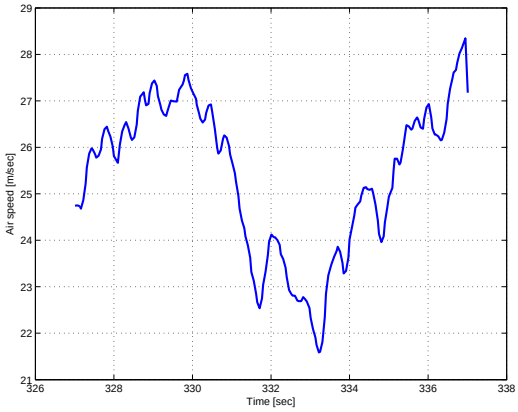
(c) Angle of attach



(d) Pitch angle

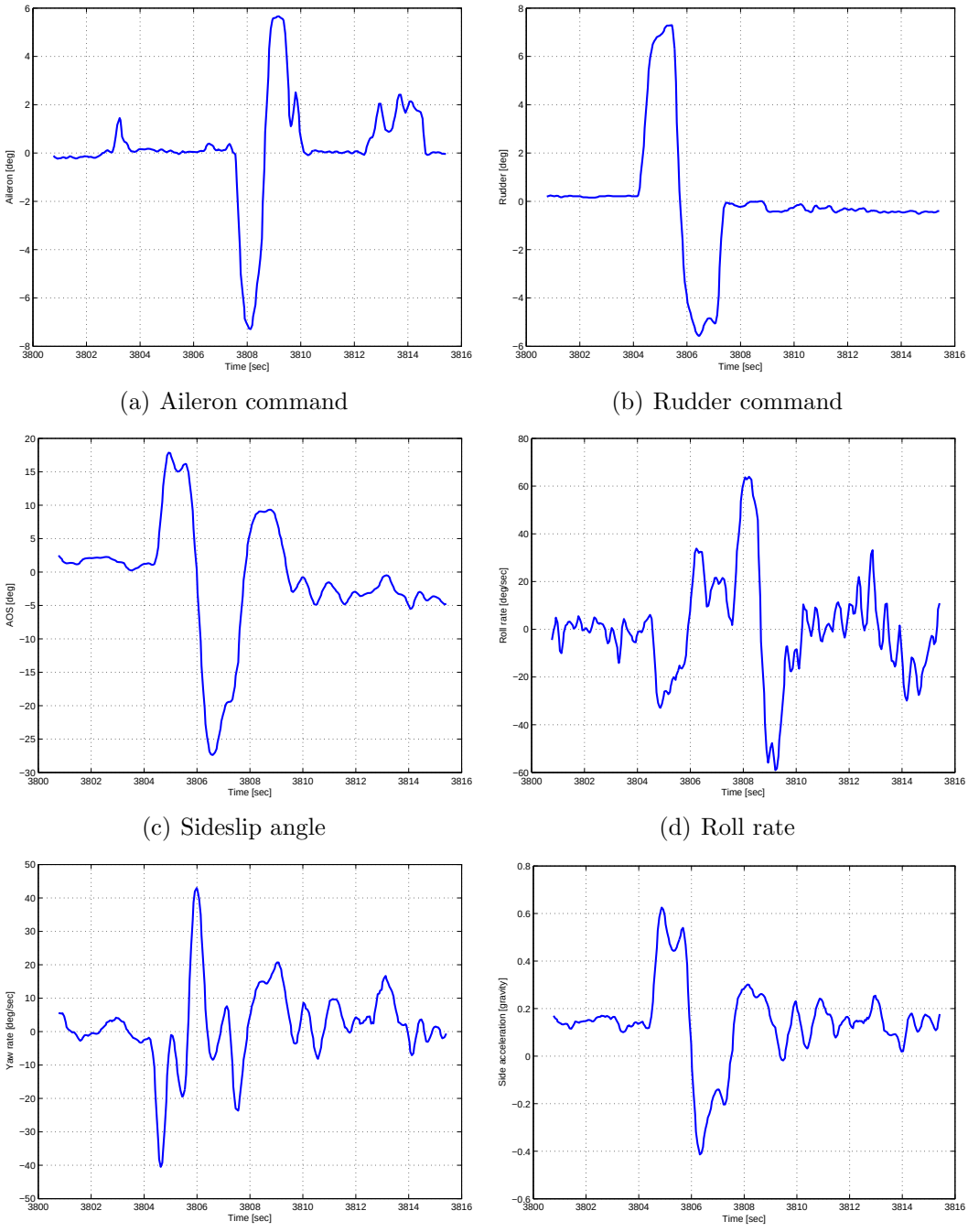


(e) Pitch rate



(f) Air speed

Figure 80: Measured state variables during a longitudinal maneuver.



(a) Aileron command

(b) Rudder command

(c) Sideslip angle

(d) Roll rate

(e) Yaw rate

(f) Side acceleration

Figure 81: Measured state variables during a lateral maneuver.

Table 14: Body x-axis aerodynamic force coefficient (C_X) identification results.

Estimates [Error, %]	C_{x_0}	C_{x_q}	$C_{x_{\delta_e}}$
Case 1	-9.441e-3 [3.1]	-13.64 [7.9]	-0.5413 [7.4]
Case 2	-9.576e-3 [3.1]	-14.66 [5.6]	-0.6207 [4.7]
Case 3	-7.331e-3 [6.1]	-19.46 [7.4]	-0.8219 [6.2]
Case 4	7.348e-3 [6.2]	-15.34 [6.3]	-0.6543 [5.3]
Case 5	-7.747e-3 [5.6]	-17.38 [5.7]	-0.7881 [4.7]

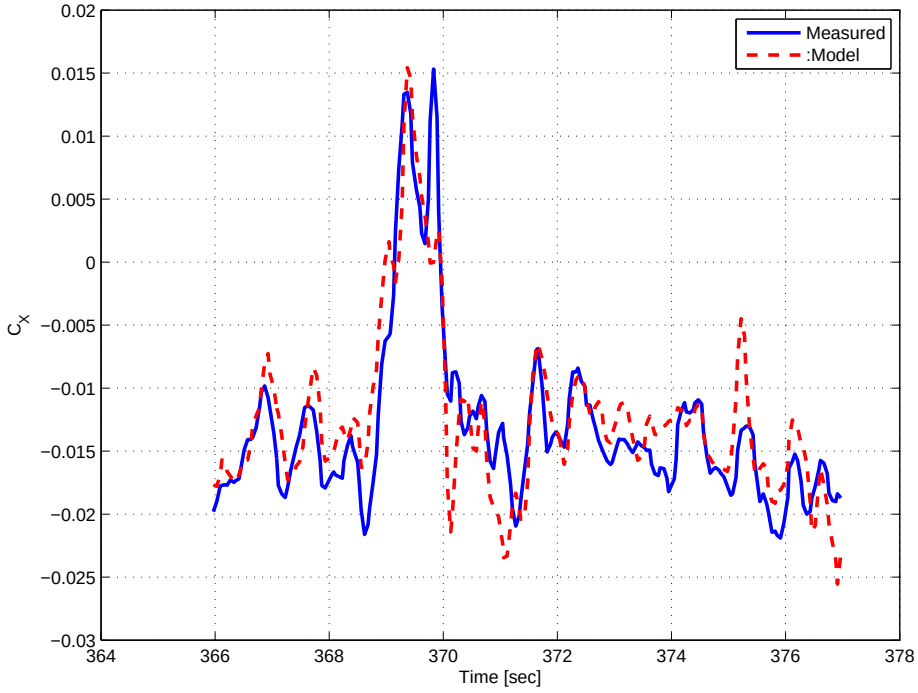


Figure 82: Validation for the force coefficient C_X.

Table 15: Body z-axis aerodynamic force coefficient (C_Z) identification results.

Estimates [Error, %]	C_{z_0}	C_{z_α}	C_{z_q}	$C_{z_{\dot{\alpha}}}$
Case 1	-0.2522 [2.3]	-2.746 [8.0]	-42.03 [9.0]	81.3 [4.9]
Case 2	-0.2588 [1.4]	-2.874 [5.9]	-43.94 [7.4]	77.91 [4.7]
Case 3	-0.2670 [1.2]	-2.848 [4.6]	-43.86 [5.7]	63.4 [4.7]
Case 4	-0.2309 [3.0]	-2.561 [9.3]	-49.08 [9.3]	64.67 [8.9]
Case 5	-0.2609 [1.4]	-3.173 [5.2]	-45.84 [8.6]	72.71 [5.9]

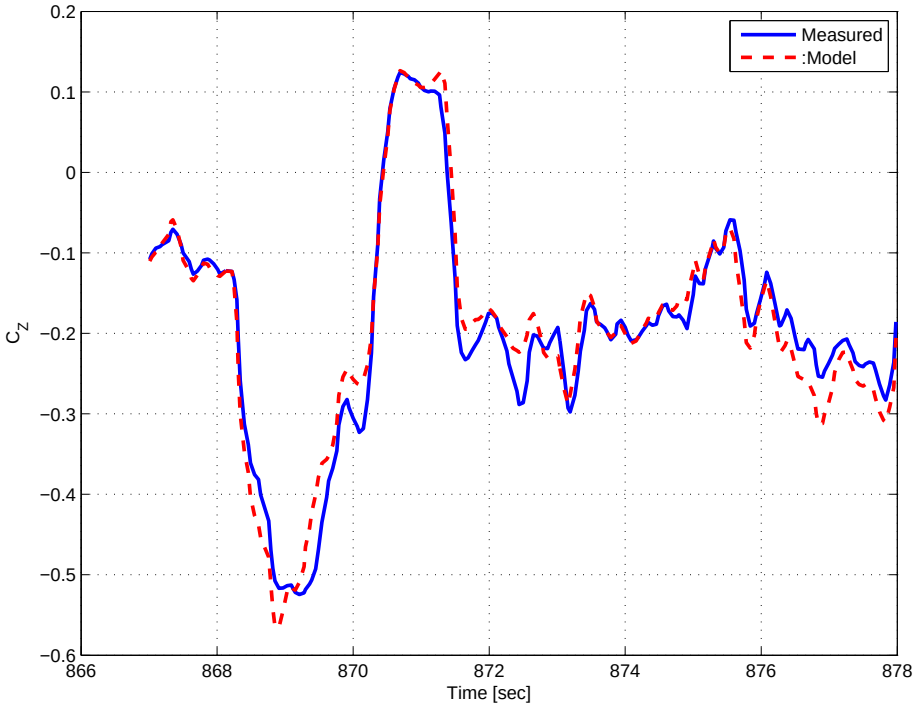


Figure 83: Validation for the force coefficient C_Z.

Table 16: Body y-axis aerodynamic moment coefficient (C_m) identification results.

Estimates [Error, %]	C_{m_0}	C_{m_α}	C_{m_q}	$C_{m_{\dot{\alpha}}}$	$C_{m_{\delta_e}}$
Case 1	-1.09e-3 [13.1]	-0.1963 [5.1]	-3.962 [13.1]	4.941 [4.6]	-0.2639 [7.4]
Case 2	-1.58e-3 [12.6]	-0.2073 [6.6]	-3.758 [17.6]	5.069 [6.0]	-0.2442 [9.8]
Case 3	-2.15e-3 [14.7]	-0.2757 [6.4]	-2.664 [18]	4.281 [8.2]	-0.2063 [7.2]
Case 4	-1.46e-3 [28.3]	-0.2636 [6.0]	-4.323 [14.7]	5.224 [5.4]	-0.244 [6.3]
Case 5	-5.80e-4 [104.8]	-0.1976 [9.6]	-4.567 [19.5]	5.747 [6.2]	-0.2086 [11.2]

Table 17: Body y-axis aerodynamic force coefficient (C_Y) identification results.

Estimates [Error, %]	C_{y_0}	C_{y_β}	C_{y_p}	C_{y_r}
Case 1	-2.645e-2 [2.8]	-0.1275 [5.7]	1.144 [10.1]	1.039 [10.2]
Case 2	-2.946e-2 [1.9]	-0.1485 [4.7]	0.5335 [14.0]	0.8015 [12.5]
Case 3	-2.566e-2 [3.6]	-0.1825 [3.7]	0.6247 [18.6]	0.7222 [15.4]
Case 4	-2.597e-2 [4.5]	-0.1634 [5.2]	1.151 [16.1]	0.8016 [16.4]
Case 5	-2.786e-2 [3.3]	-0.1561 [4.5]	1.036 [11.7]	0.8627 [15.2]

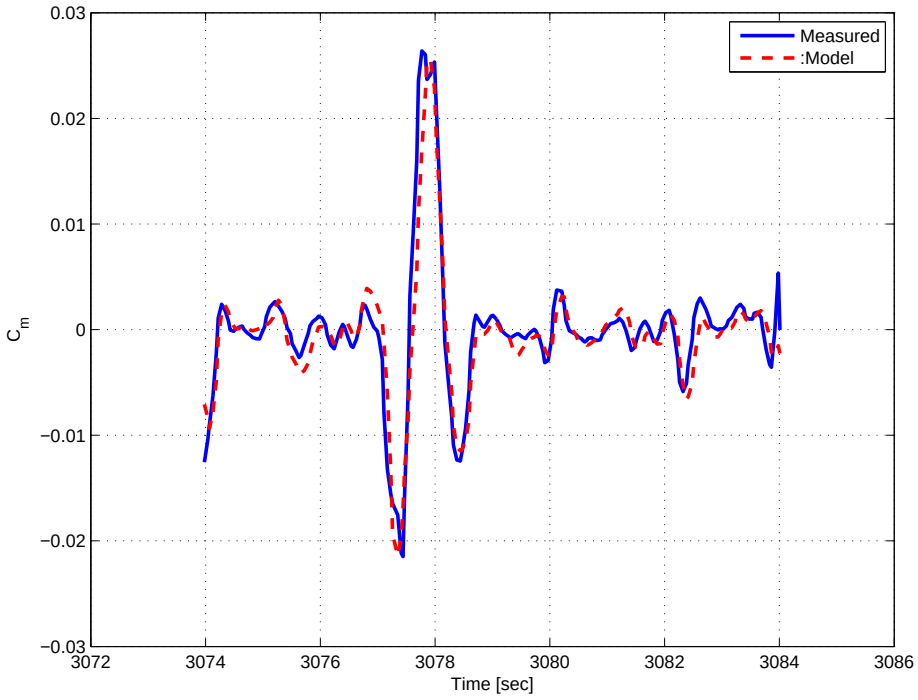


Figure 84: Validation for the moment coefficient C_m.

Table 18: Body x-axis aerodynamic moment coefficient (C_ℓ) identification results.

Estimates [Error, %]	C_{ℓ_0}	C_{ℓ_β}	C_{ℓ_p}	$C_{\ell_{\delta_a}}$
Case 1	-3.652e-4 [10.1]	-1.305e-2 [4.9]	-1.239e-2 [36.7]	-1.362e-2 [15.2]
Case 2	-2.629e-4 [17.2]	-9.19e-3 [8.2]	-2.566e-2 [25.5]	-1.517e-2 [18.5]
Case 3	3.90e-4 [11.9]	-7.61e-3 [6.6]	-3.499e-2 [19.1]	-1.752e-2 [16.8]

Table 19: Body z-axis aerodynamic moment coefficient (C_n) identification results.

Estimates [Error, %]	C_{n_0}	C_{n_β}	C_{n_p}
Case 1	-7.571e-4 [11.1]	1.179e-2 [9.4]	-6.768e-2 [15.2]
Case 2	2.180e-4 [31.4]	5.986e-3 [16.7]	-0.1077 [10.5]
Case 3	-3.244e-4 [24.9]	1.029e-2 [13.7]	-3.968e-2 [27.1]
Case 4	1.822e-4 [53.3]	1.005e-2 [13.9]	-9.096e-2 [15.7]
Estimates [Error, %]	C_{n_r}	$C_{n_{\delta_a}}$	$C_{n_{\delta_r}}$
Case 1	-0.1322 [12.3]	-3.065e-2 [14.3]	-5.691e-2 [9.1]
Case 2	-6.577e-2 [22.5]	-4.902e-2 [10.2]	-4.107e-2 [11.9]
Case 3	-8.447e-2 [20.6]	-1.712e-2 [26.8]	-3.841e-2 [14.6]
Case 4	-9.061e-2 [17.3]	-4.827e-2 [13.0]	-5.221e-2 [10.5]

linear approximation of the aerodynamic forces and moments is used to simulate realistic dynamic behavior of the aircraft. The nonlinear aircraft model also involves the detail modeling of subsystems such as sensors and actuators using experimental data. In addition, the external pilot command input and the flight visualization enable the simulation to be used as a virtual flight test.

The actual autopilot hardware is placed inside the simulation loop in order to test and validate both the hardware and the on-board software. Four independent computer systems were used in the hardware-in-the-loop (HIL) simulation as illustrated in Fig. 85: the 6-DOF simulator, the flight visualization computer, the autopilot micro-controller, and the ground station computer console. A detail description for this setup is given below.

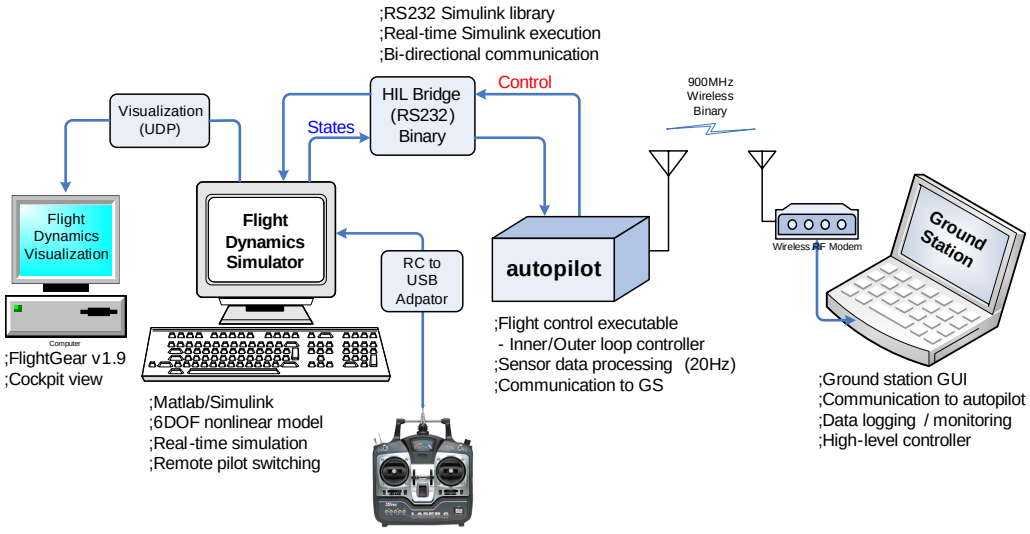


Figure 85: High fidelity hardware-in-the-loop (HIL) simulation environment.

C.4.1 A 6-DOF simulator

A full 6-DOF nonlinear model for the aircraft was built in Matlab/Simulink® environment. The 12 states differential equations of motion in Eqs. (163)-(166) are utilized in conjunction with an approximation of the aerodynamic forces and moments via component buildup. The stability and control derivatives are either obtained in

the manner as discussed in Section C.1.2 or Section C.3. The stability and control derivatives from the geometry of the aircraft can be chosen during the early phase of development, however, once these derivatives have been identified experimentally, the identified values are used for realistic simulation results. The 6-DOF nonlinear model becomes more realistic by employing a standard atmospheric model, an Earth and gravity model (WGS-84), and an Earth magnetic field model[144]. The output states from the simulator are processed to emulate real sensors accounting for sensor latency, random walk bias, and measurement noise. After digitized according to the word size of the micro-controller (12 bit, 4096 steps), the sensor values are transmitted to the autopilot via a serial communication. In a similar manner, the navigation states are processed for data latency and measurement noise and encapsulated in a binary GPS packet identical to the actual GPS sensor output. This packet is also transmitted to the autopilot at low update rate (1 Hz) via the serial communication.

In the HIL environment, the autopilot functions identically as in a real flight test, while computing control commands to each control actuator. These commands are sent back to the 6-DOF simulator in order to drive the actuator model of the control surface. Each control surface was modeled as a first order system with a rate limiter and saturation limit, whose parameters were identified experimentally. Consequently, a simulation loop is formed between the 6-DOF simulator (software) and the autopilot (hardware) exchanging the simulated data back and forth.

C.4.2 User interface

An R/C transmitter is connected to the 6-DOF simulator for recording remote pilot stick commands: Four channels for control surface commands ($\delta_a, \delta_e, \delta_t$ and δ_r) and two auxiliary commands for toggling between autonomous control mode and remote pilot mode. The remote pilot stick commands can override the autopilot control command at any time by switching the commands to the actuator models at user's

choice. In this case, a simulation for an open loop maneuver can be conducted along the remote pilot input to validate the dynamic characteristic of the aircraft model.

In order to visualize the simulation, we adopted the use of FlighGear[1], an open-source flight simulator for a visualization tool. The FlightGear is a flight simulator framework, which has been widely used in various research environment[107, 127, 36, 133]. Although it provides a total simulation environment in conjunction with the internal flight dynamics models, we used the FlightGear as a visualization tool combining with the 6-DOF simulator already developed. By doing this, the FlightGear receives the simulated states from the simulator at a fixed update rate to constantly refresh the virtual scenes that have been generated along the user's convenient view angles.

The ability to record the pilot stick command with visualizing via the FlightGear allows the simulation environment to replace a real experiment to save time and effort. An open loop behavior of the UAV can be tested and validated by a remote pilot via the user friendly input device, and the closed loop control performance can be verified and demonstrated by conducting virtual experiments in advance to the real flight test. Figure 86 shows a computer screen shot taken during the HIL simulation visualizing the dynamic behavior of the UAV, while displaying the corresponding state variables in the scopes.

C.4.3 Data communication and synchronization

The 6-DOF simulator exchanges the sensor and control command signals with the autopilot via a serial communication between the two. The sensor output data include 16 different sensor outputs from rate gyros, accelerometers, magnetometers, and etc., and are packed into a custom-designed binary packet. The binary packet begins with a designated header to distinguish between different packets and ends by a trailer for data consistency check. The serial communication is configured for a

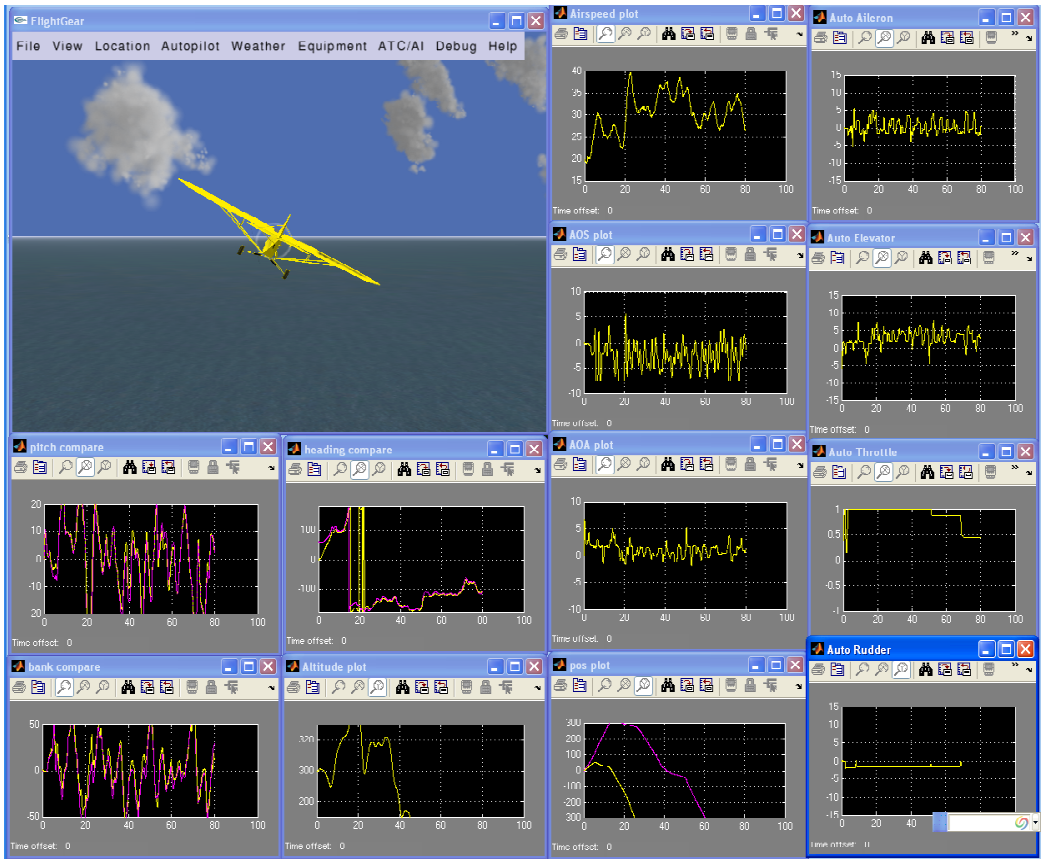


Figure 86: Hardware-in-the-loop simulation screen shot

baud rate 115200 bps, which allows communicating by transmitting the sensor packet of 39 bytes at 20 Hz as well as the GPS packet of 54 bytes at 1 Hz update rate. The control command data from the autopilot are composed of four PWM commands for each actuator, resulting in a binary packet of 11 bytes in length associated with header and trailer to be transmitted at 20 Hz update rate. The HIL bridge shown in Fig. 86 was then incorporated into the 6-DOF Simulink model to handle the bidirectional communication with a help of the serial communication block toolbox[82]. In addition, a dedicated S-function block is utilized to send simulated output to the FlightGear[144]. The S-function block bundles the simulated data with a user defined protocol (UDP) for small size of a data packet to ensure less communication overhead over the network. Real time visualization is then made possible unless the network latency is significant. To minimize the network latency, one should locate two hosts

(for 6-DOF simulation and 3-D visualization) within a local area network (LAN), or if the computer resources permits one can configure to use a single computer for visualizing and running the 6-DOF simulation on the same machine.

Because the multiple systems are involved in the HIL simulation, it is all important to keep the execution timing among these systems properly synchronized during the entire simulation period for data compatibility among them. Besides, a real-time execution is necessary for the simulation to be in accord with the pilot input and the visualization at correct time step. Although the 6-DOF simulator was written in a Simulink model running at non real-time, a real time execution was accomplished by utilizing the Realtime block toolbox[83] in the simulation. This toolbox is based on the simple concept that if the current hardware cycle time is lower than the desired simulation step time, the block simply hold the current execution until the desired time step is triggered. For a real time simulation, the basic simulation time step of the 6-DOF simulator is carefully chosen at 100 [Hz] taking into consideration both the computational overhead and the faithful simulation results for accurate dynamic characteristics of the UAV. A different sampling rate at 20 [Hz] was utilized for the serial communication block and the FlightGear S-function block. The synchronization of the data was implicitly dealt with in a manner that the main simulation loop of the 6-DOF simulator polls each communication block at a fixed interval during the simulation, whereas the autopilot and the FlightGear receive the packets passively.

C.5 Summary

A hardware-in-the-loop simulation environment has been built to validate hardware and software development of the autopilot for a small UAV. A full 6-DOF nonlinear dynamic model has been used in conjunction with the linear approximation of the aerodynamic forces and moments. A batch parameter identification method has been used to determine the stability and control derivatives of the UAV using flight

test data. Detailed models for the sensors and actuators were incorporated in the simulation along with the capability of real-time 3-D visualization and external pilot interface. The hardware-in-the-loop simulation is an indispensable tool for performing simulated flight tests in support of avionics development and validation of control laws for small UAVs with minimal cost and effort.

APPENDIX D

DESIGN OF INNER CONTROL LOOPS

D.1 Decoupled linear model

For the design purpose of inner loop controls of the UAV, we employ a decoupled linear model, which has been built from stability and control derivatives. The stability and control derivatives of the UAV utilized in this research have been experimentally identified through the approach described in Appendix C. The resulting linear model is composed of two linear systems which describe the longitudinal and lateral dynamics of the UAV:

Longitudinal dynamics

$$\dot{x}_{\text{Long}} = A_{\text{Long}} x_{\text{Long}} + B_{\text{Long}} u_{\text{Long}}, \tag{190}$$

where the state vector x_{Long} and the control input u_{Long} are defined by

$$x_{\text{Long}} = [\Delta\alpha \ \Delta q \ \Delta v_T \ \Delta\theta]^{\mathsf{T}}, \quad u_{\text{Long}} = [\Delta\delta_e \ \Delta\delta_t]^{\mathsf{T}},$$

and the symbol Δ denotes a perturbed representation of the corresponding variables. The system matrices were obtained from the stability and control derivatives,

$$A_{\text{Long}} = \begin{bmatrix} -7.1264 & 0.9497 & -0.0485 & 0.0 \\ -122.8676 & -4.2920 & -0.0261 & 0.0 \\ 4.1512 & -0.0398 & -0.1935 & -9.81 \\ 0 & 1.0000 & 0.0 & 0.0 \end{bmatrix}, \tag{191a}$$

$$B_{\text{Long}} = \begin{bmatrix} -0.2953 & -0.0174 \\ -83.4069 & 0.8221 \\ -1.3732 & 6.1044 \\ 0.0 & 0 \end{bmatrix}. \tag{191b}$$

Lateral dynamics

$$\dot{x}_{\text{Lat}} = A_{\text{Lat}} x_{\text{Lat}} + B_{\text{Lat}} u_{\text{Lat}}, \tag{192}$$

where the state vector x_{Lat} and the control input u_{Lat} are defined by

$$x_{\text{Lat}} = [\Delta\beta \ \Delta\phi \ \Delta p \ \Delta r]^{\mathsf{T}}, \quad u_{\text{Lat}} = [\Delta\delta_a \ \Delta\delta_r]^{\mathsf{T}},$$

and the system matrices were obtained from the identified stability and control derivatives as follows,

$$A_{\text{Lat}} = \begin{bmatrix} -0.3301 & 0.4897 & 0.0570 & -0.9939 \\ 0.0 & 0.0 & 1.0 & 0.0570 \\ -35.4688 & 0.0 & -16.8528 & 0.1524 \\ 2.7315 & 0.0 & -1.1105 & -0.5340 \end{bmatrix}, \tag{193a}$$

$$B_{\text{Lat}} = \begin{bmatrix} 0.0 & 0.1494 \\ 0.0 & 0 \\ -182.2395 & 9.2613 \\ -9.4423 & -7.7841 \end{bmatrix}. \tag{193b}$$

The servo actuators were modeled as first-order systems with a corner frequency of 10 [rad/sec] as identified experimentally. By convention, the positive control surface deflections are supposed to generate negative aerodynamic moments about each corresponding axis (roll moment by aileron, pitch moment by elevator, and yaw moment by rudder). In order to use a negative feedback structure of the classical control design method, we define different control inputs u_a, u_e, and u_r for aileron, elevator, and rudder, respectively. Subsequently, we take into account not only the actuator dynamics but also the sign convention of control inputs to come up with the actuator

models as follows,

$$\Delta\delta_a = \tfrac{-10}{s+10}\Delta u_a, \tag{194a}$$

$$\Delta\delta_e = \tfrac{-10}{s+10}\Delta u_e, \tag{194b}$$

$$\Delta\delta_t = \tfrac{10}{s+10}\Delta u_t, \tag{194c}$$

$$\Delta\delta_r = \tfrac{-10}{s+10}\Delta u_r. \tag{194d}$$

D.2 Longitudinal controllers design

In this section, we describe a detail design of inner loop controllers for the longitudinal dynamics of the UAV. The longitudinal controllers comprise a pitch controller and a speed controller, which calculate the elevator command input (Δu_e) and the throttle command input (Δu_t), respectively.

D.2.1 Pitch controller

We design the pitch controller starting from a pitch damper. The open-loop transfer function from the elevator command input (Δu_e) to the pitch rate (Δq) is obtained from Eqs. (191) and (194b) as follows,

$$\frac{\Delta q}{\Delta u_e} = \frac{834.07s(s+6.645)(s+0.2394)}{(s+10)(s+5.721\pm 10.706i)(s+0.085\pm 0.6141i)}, \tag{195}$$

which reveals the natural frequency of the phugoid mode is 0.6199 [rad/sec] (damping ratio is 0.1371) and the natural frequency of the short period mode is 12.1391 [rad/sec] (damping ratio is 0.4713). It also shows the natural frequency of the servo actuator by 10 [rad/sec] (damping ratio is 1.0). Figure 87 shows a root locus plot of this transfer function with a unity negative feedback of the pitch rate with a positive gain k_q, which shows that the short period poles get poorly damped as k_q increases towards infinity. In order to improve the damping characteristics of the short period mode, a lead compensator is added in the feedback path with a compensator pole of $s = -16$ to minimize the influence of the compensator pole on the open loop plant.

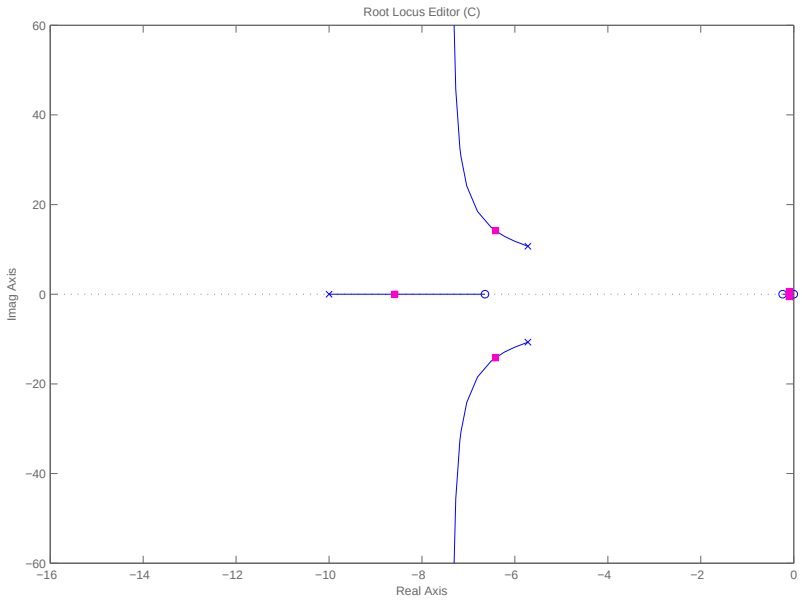


Figure 87: Root locus plot of the transfer function in Eq. (195) with a unity negative feedback of the pitch rate Δq with the positive gain k_q. As the gain increases, the short period mode gets poorly damped.

The pole/zero ratio is carefully chosen by eight, hence the compensator is given by

$$G_q(s) = k_q \frac{s+2}{s+16}, \tag{196}$$

where the compensator gain is chosen $k_q = 0.101$ in order for the short period mode damping to be close to the critical damping ratio. The closed loop transfer function from a commanded pitch rate (Δq_r) to the pitch rate (Δq) after closing the lead compensator is calculated as follows,

$$\frac{\Delta q}{\Delta q_r} = \frac{834.07s(s+16)(s+0.2394)(s+6.645)}{(s+9.145 \pm 12.196i)(s+9.578 \pm 3.844i)(s+0.083 \pm 0.599i)}. \tag{197}$$

Figure 88 shows the root locus plot incorporating the designed lead compensator in the feedback path. With the lead compensator, the phugoid mode does not change much, yet the short period mode become properly damped as resulting in the natural frequency of 15.2441 [rad/sec] and the damping ratio of 0.6.

Having designed the pitch damper, we design a pitch attitude controller that forms an outer loop of the system given in Eq. (197). To this end, a transfer function from

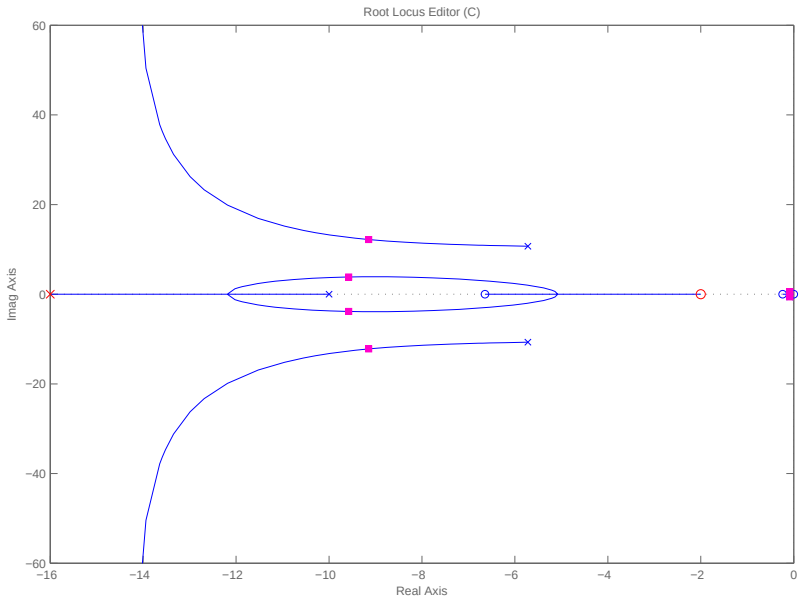


Figure 88: Root locus plot of the closed-loop system incorporating the lead compensator of the pitch rate. The closed-loop poles and zeros are shown, and the squares represent the location of the poles at $k_q = 0.101$.

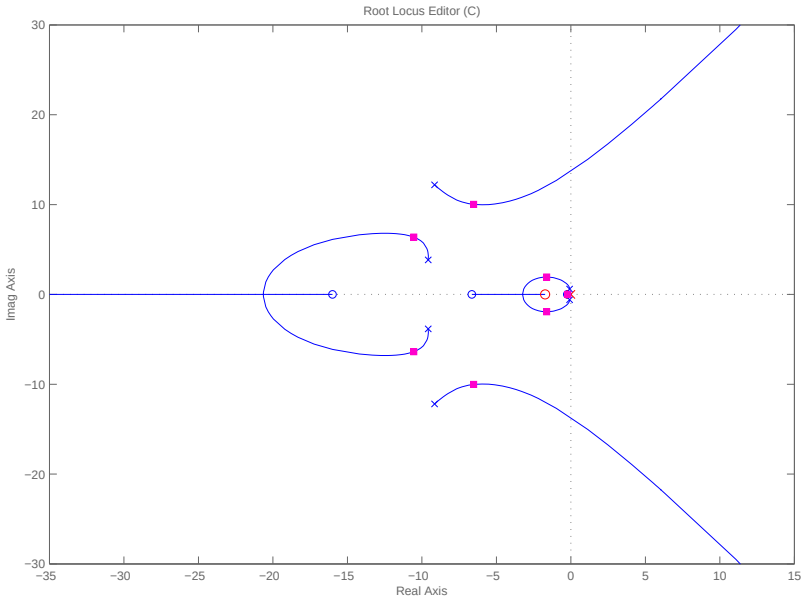
the commanded pitch rate (Δq_r) to the pitch angle ($\Delta\theta$) is obtained by

$$\frac{\Delta\theta}{\Delta q_r} = \frac{834.07(s+16)(s+0.2394)(s+6.645)}{(s+9.145\pm 12.196i)(s+9.578\pm 3.844i)(s+0.083\pm 0.599i)}. \tag{198}$$

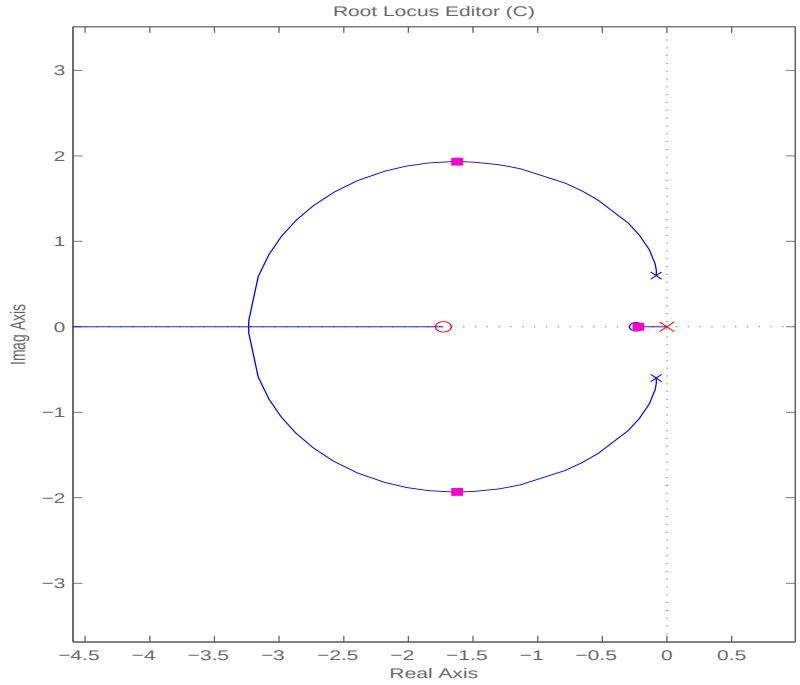
The transfer function appears to be a *Type-0* system that has no integrator pole in the characteristic equation. Hence, in order to remove a steady state error of the pitch angle we design a proportional integral (PI) controller. The PI controller with the zero located at $s = -1.73$ is given as follows,

$$G_\theta(s) = k_\theta \frac{s+1.73}{s}, \tag{199}$$

yields the phugoid poles are located at $s \approx -1.6 \pm 1.9i$ of the closed-loop system with the proportional gain $k_\theta = 0.855$, which provides a critical damping of phugoid mode (see Fig. 89(b) for the root locus plot of the closed-loop system). Figure 89(a) shows the entire root locus plot of the closed-loop system, where the short period poles are also properly damped. In addition, the Bode plot of the closed-loop system is displayed in Fig. 90, which shows that the closed-loop system has the gain margin of 12.4 [dB] and the phase margin of 51.6 [deg].



(a) Entire display of the root locus plot of the closed-loop system.



(b) Magnified display of the root locus plot showing the phugoid poles around the origin.

Figure 89: Root locus plot of the closed-loop system with the pitch angle PI controller as k_θ varies. The squares represent the closed-loop poles at $k_\theta = 0.855$.

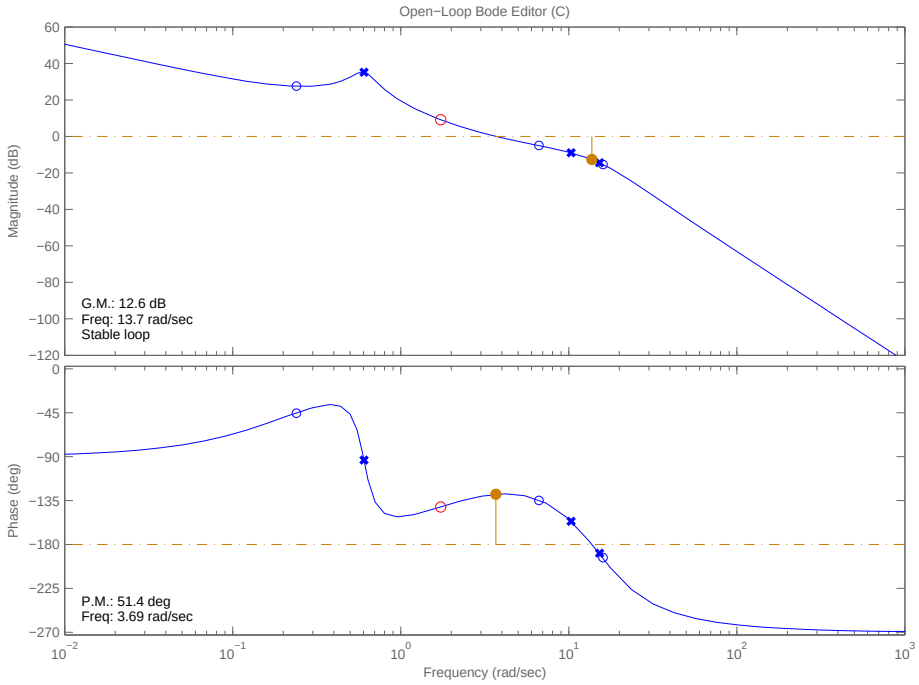


Figure 90: Bode plot of the closed-loop system with the pitch angle PI controller.

The final closed-loop transfer function from the pitch angle reference command ($\Delta\theta_r$) to the pitch angle ($\Delta\theta$) becomes,

$$\frac{\Delta\theta}{\Delta\theta_r} = \frac{713.13(s+0.24)(s+1.73)(s+6.65)(s+16)}{(s+0.22)(s+1.67\pm1.94i)(s+6.41\pm10.01i)(s+10.62\pm6.42i)}, \qquad (200)$$

which reveals that the natural frequency of the phugoid mode ends up with 2.5539 [rad/sec] (damping ratio 0.6541), and the natural frequency of the short period mode becomes 11.8813 [rad/sec] (damping ratio 0.5392). Figure 91 illustrates the final implementation of the pitch attitude controller including the pitch damper. The proportional term of the PI controller is fed from the feedforward signal $\Delta\theta_e$ as drawn by a dashed line and is added together with integral term. A step response of the closed-loop system by a unit pitch reference command is shown by a dashed line in Fig. 92. The performance of the PI controller is given by an overshoot 120% and a zero steady state error with the settling time of 2.9 [sec].

By the PI implementation discussed above, the integrator pole of the controller introduces an additional zero in the closed-loop system. Excessive overshoot arises

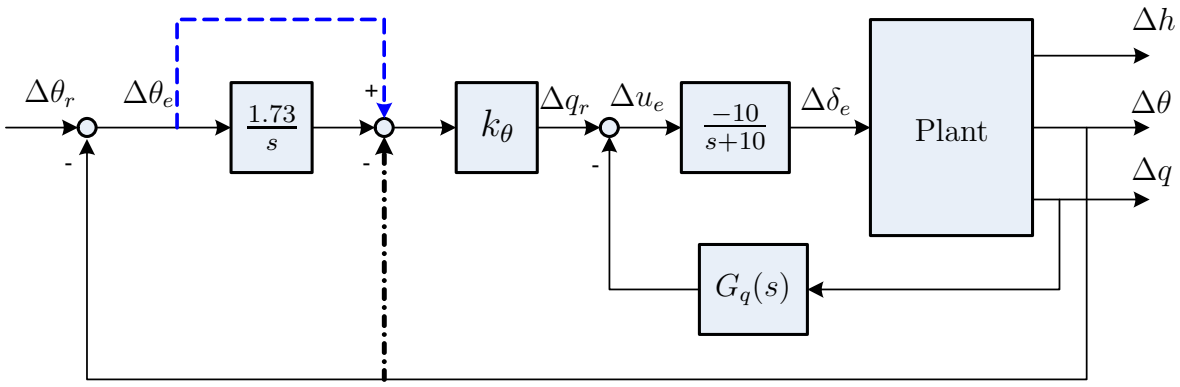


Figure 91: Block diagram of the closed-loop pitch angle controller. The dashed line denotes the original PI controller implementation, whereas the dashed-dot line denotes an alternative PI implementation with closed-loop zero removed.

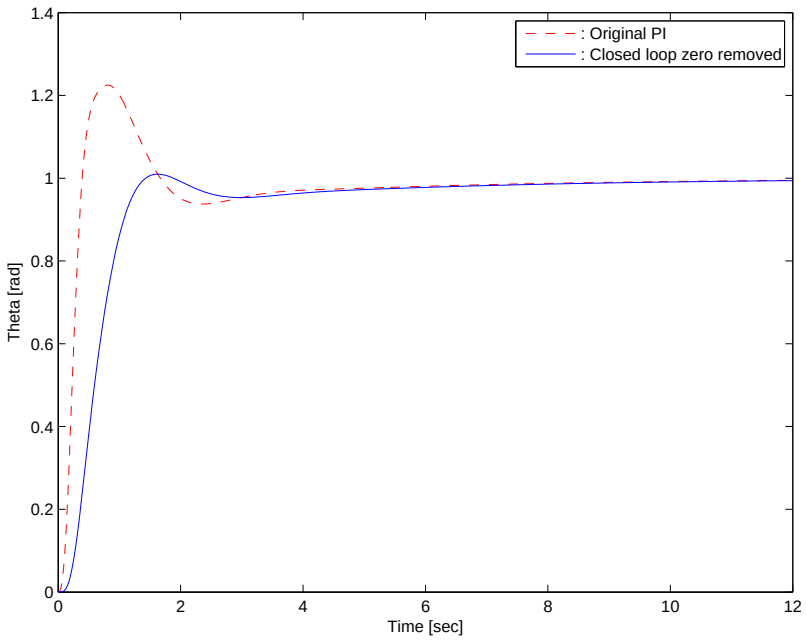


Figure 92: Step response of the closed-loop system by a unit pitch reference command.

from this closed-loop zero, thus degrading the overall performance (see Fig.92 for the dashed response). On the other hand, an alternative PI implementation which does not induce a closed-loop zero can be incorporated in the manner that the proportional term of the PI is fed from the feedback signal of $\Delta\theta$[137] as shown in Fig. 91 by a dashed-dot line. By this implementation, it is easy to verify that the characteristic equation of the closed-loop remains same, while the zero induced from the integrator pole is removed from the closed-loop system. The step response of this implementation is shown in Fig. 92, where the overshoot of the step response is reduced at the cost of slow rise time while attaining approximately same settling time. Consequently, since the latter PI implementation (no closed-loop zero) results in the tight control bound with a smooth transition, it is likely to be a better choice for the actual implementation.

D.2.2 Altitude Controller

After closing the inner loop pitch controllers discussed in the previous section, we obtain an open-loop plant for the purpose of altitude controller design. Incorporating the PI controller implementation that removes the closed-loop zero, we obtain a transfer function from the pitch reference angle command ($\Delta\theta_r$) to the altitude (Δh) as follows,

$$\frac{\Delta h}{\Delta\theta_r} = \frac{-87.37(s+0.14)(s+16)(s+39.42)(s-49.59)}{s(s+0.20)(s+1.76\pm1.93i)(s+6.44\pm9.95i)(s+10.51\pm6.61i)}. \quad (201)$$

It follows from Eq. (201) that due to a non-minimum zero in the transfer function, it is anticipated that an excessive proportional feedback gain might cause instability of the system. Hence, in order to improve the performance of the transient response while ensuring the stability of the system, a lead compensator is adopted as follows,

$$G_{hL}(s) = 0.3\frac{s+2.5}{s+20}, \quad (202)$$

which results in the gain margin 13.1 [dB] and the phase margin 79.8 [deg] of the closed-loop system. The lead compensator improves the transient response, however,

it reduces the low frequency gain by -27 dB. As a result, a steady-state error to a unit ramp input will grow accordingly. In order to boost the low frequency gain and thus reduce the steady-state error, a PI compensator is adopted, albeit it adds small phase lag in the higher frequency range, as follows

$$G_h(s) = k_h \frac{(s + 0.495)}{s}, \tag{203}$$

where the proportional gain of the PI controller is chosen $k_h = 1.22$.

Figure 94 is the root locus plot of the altitude feedback using the lead and the PI compensators as k_h varies. With the chosen gain $k_h = 1.22$, the final closed-loop transfer function from the altitude reference command (Δh_r) to the altitude (Δh) yields,

$$\frac{\Delta h}{\Delta h_r} = \frac{-31.98(s + 0.14)(s + 0.495)(s + 2.5)(s + 16)(s + 39.42)(s - 49.59)}{s(s + 0.15)(s + 0.54 \pm 0.51i)(s + 1.0 \pm 1.96i)(s + 6.49 \pm 10.14i)(s + 10.71 \pm 6.49i)(s + 19.99)}, \tag{204}$$

where the phugoid poles are located at $s = -0.54 \pm 0.51i$, which gives the natural frequency of 0.742 [rad/sec] with the damping ratio of 0.731. The gain and phase margin plot of the closed-loop system is shown in Fig. 93, giving the margins by 9.31 [dB] and 45 [deg], respectively. From the step response shown in Fig. 95, the closed-loop system has a fast rise time but an undesirable overshoot. It follows from the arguments in the previous section, the excessive overshoot is caused by the induced closed-loop zero due to the PI controller. This overshoot is then relaxed by implementing the other PI controller, where the altitude signal is fed back before/after the the integrator block. The closed-loop system ends up with the same closed-loop characteristic equations yet the closed-loop zero removed. Hence, as shown in Fig. 95, the overshoot performance is improved by 110% compared to 130% with no closed-loop zero removal, by sacrificing the transient performance with the increased rise time about 2.5 [sec].

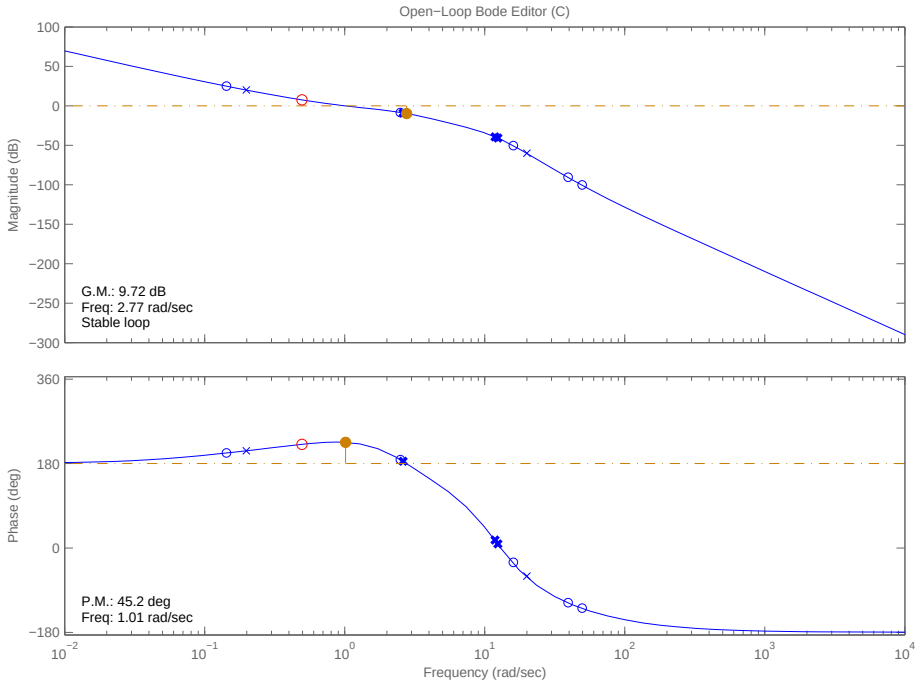
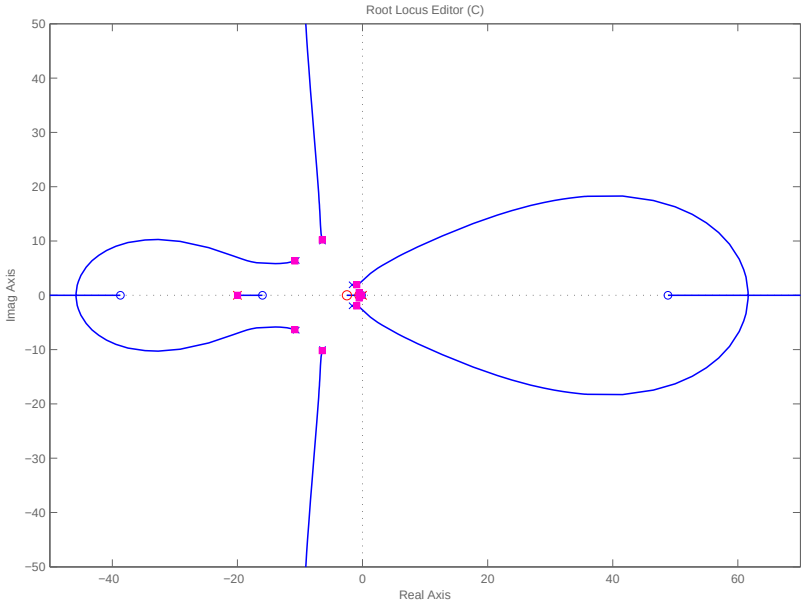


Figure 93: Bode plot of the closed-loop system with the lead compensator.

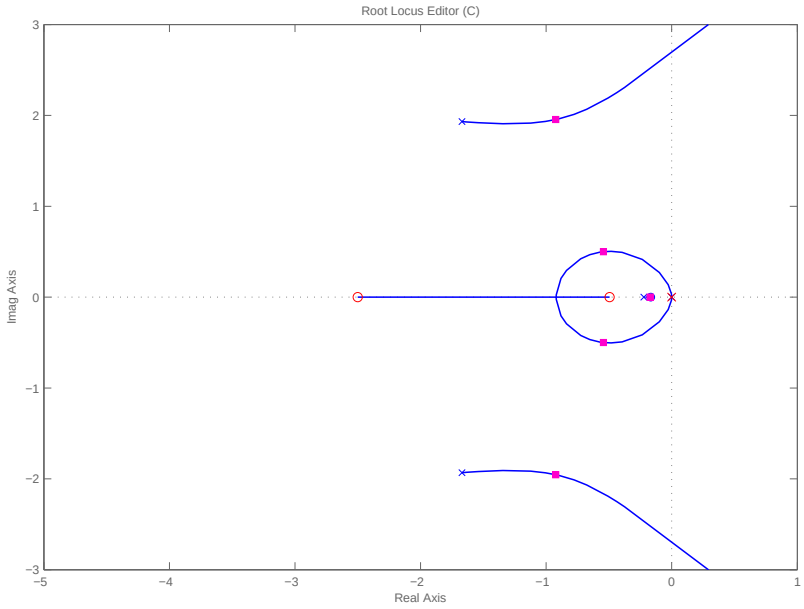
D.2.3 Integrator antiwindup

The altitude controller computes the pitch reference command for the inner loop pitch angle controller. In order to prevent the reference command from exceeding the acceptable range of the pitch controller, we put a saturation block after the altitude controller. The saturation limit for the pitch reference is chosen as $|\Delta\theta_{c_{\max}}| < 30$ [deg], thus the actual pitch angle of the vehicle is controlled within a linear region of operation. Because we introduced a voluntary saturation limit to the output of the altitude controller, the integral control action of the altitude controller might be saturated whenever a large altitude error occurs. Subsequently, the accumulated integral error terms cannot be removed until when the altitude error gets small, resulting in a substantial overshoot.

The solution to this problem is the integrator antiwindup, which turns off the integral action as soon as the saturation occurs. Figure 96 illustrates a block diagram that implements the integrator antiwindup technique using a nonlinearity. As soon



(a) Entire display of the root locus plot of the altitude closed-loop system.



(b) Magnified display of the root locus plot showing the phugoid poles around the origin.

Figure 94: Root locus plot of the closed-loop system using a lead and PI compensators as k_h varies. The squares represent the closed-loop poles at $k_h = 1.22$.

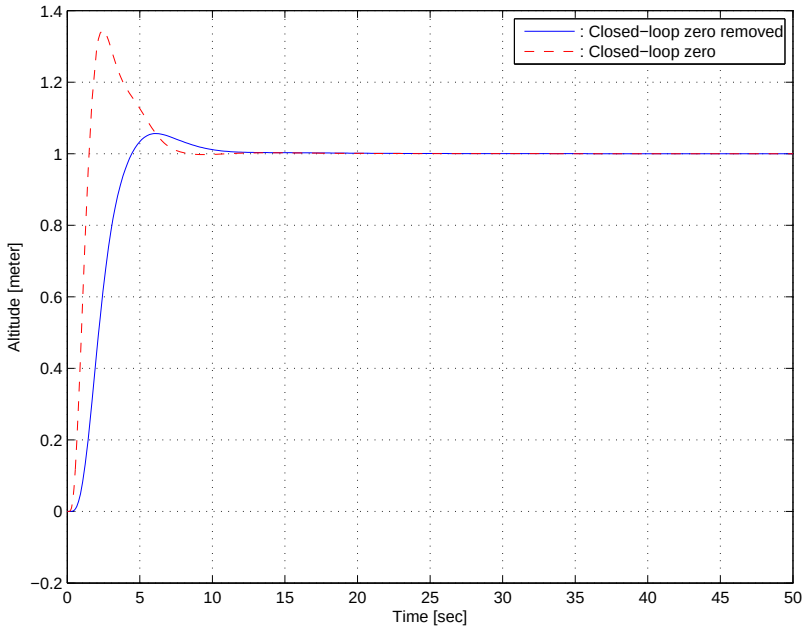


Figure 95: Step response of the closed-loop system by a unit altitude reference command.

as the saturation occurs, the feedback loop around the integrator reacts rapidly to keep e_1 at zero. The antiwindup gain K_a should be chosen to be large enough so that the antiwindup circuit is capable of following e while keeping the output from saturating.

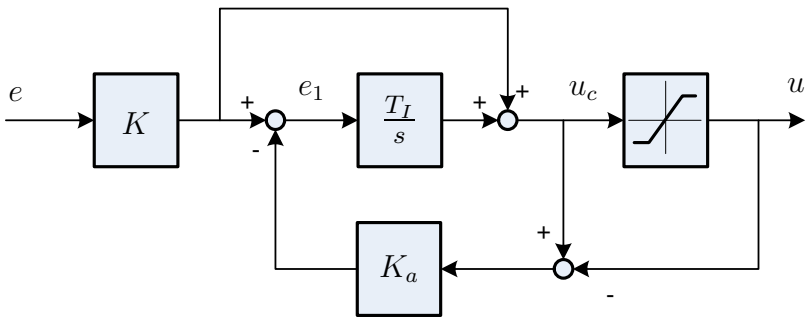


Figure 96: Integrator antiwindup technique with a single saturation nonlinearity.

Figure 97 shows the entire block diagram of the closed-loop altitude controller incorporating the integrator antiwindup scheme. Figure 98 shows the step response of the closed-loop system with/without the antiwindup scheme. It should be noted that the altitude controller implemented with the antiwindup element has substantially

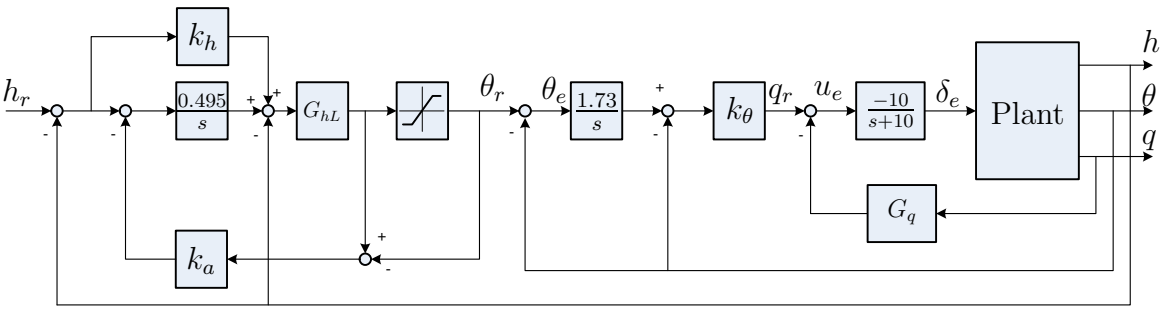


Figure 97: Block diagram of the final closed-loop altitude controller.

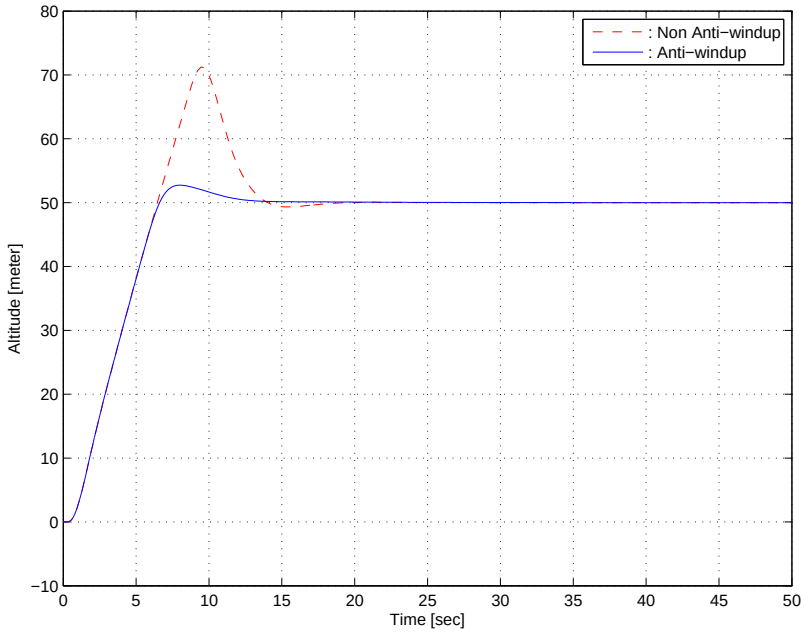


Figure 98: Step response of the closed-loop system with the altitude controller, comparing the cases of with/without antiwindup scheme.

less overshoot compared to the response of the original PI controller implementation.

D.2.4 Airspeed controller

An open-loop transfer function from the throttle command input (Δu_t) to the airspeed (ΔV_T) is obtained including the actuator dynamics using Eqs. (191) and (194c) as follows,

$$\frac{\Delta V_T}{\Delta u_t} = \frac{61.0413(s+0.07)(s+5.67\pm 10.75i)}{(s+10)(s+0.09\pm 0.64i)(s+5.72\pm 10.71i)}. \tag{205}$$

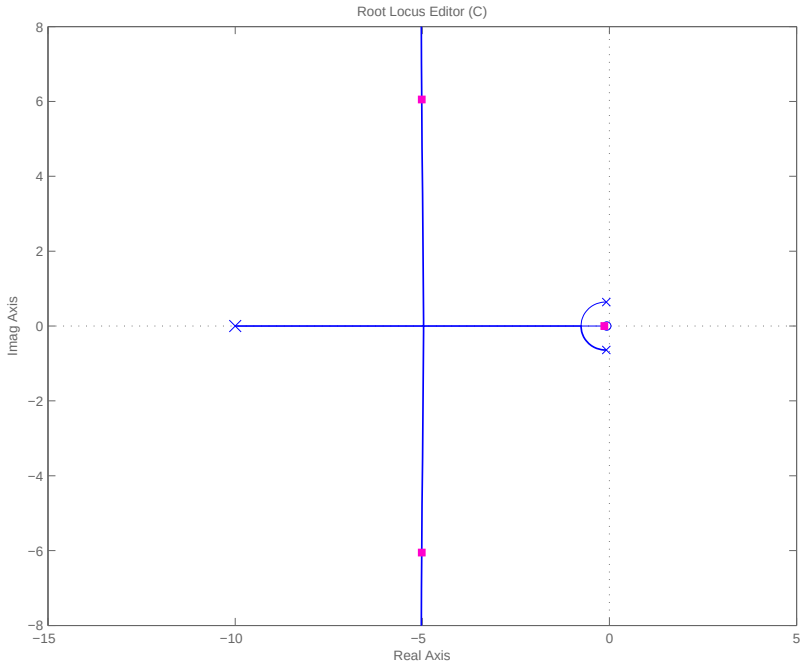


Figure 99: Root locus plot of the simplified speed control plant

Since the short period mode poles appears to be located close to the complex zeros, they could be cancelled out for the sake of simplicity. The simple transfer function is then given,

$$\frac{\Delta V_T}{\Delta u_t} = \frac{61.0413(s+0.07)}{(s+10)(s+0.09 \pm 0.64i)}. \tag{206}$$

As shown in the root locus plot (see Fig. 99), due to the zero close to the origin ($s = -0.07$), the DC gain of the plant is relatively small,

$$K_p = \lim_{s \to 0} \frac{\Delta V_T(s)}{\Delta u_t(s)} = 1.023, \tag{207}$$

which results in a large steady state error $1/(1 + K_p) \approx 0.5$ when using the unity negative feedback scheme. In order to reduce the steady state error while improving the performance of the speed controller, a PI controller was adopted with PI zero at $s = -1.53$,

$$G_{V_T}(s) = k_{V_T} \frac{s + 1.53}{s}. \tag{208}$$

Figure 100 shows the root locus plot of the closed-loop system with the airspeed controller. The PI zero at $s = -1.53$ in conjunction with the chosen gain $k_{V_T} = 0.773$

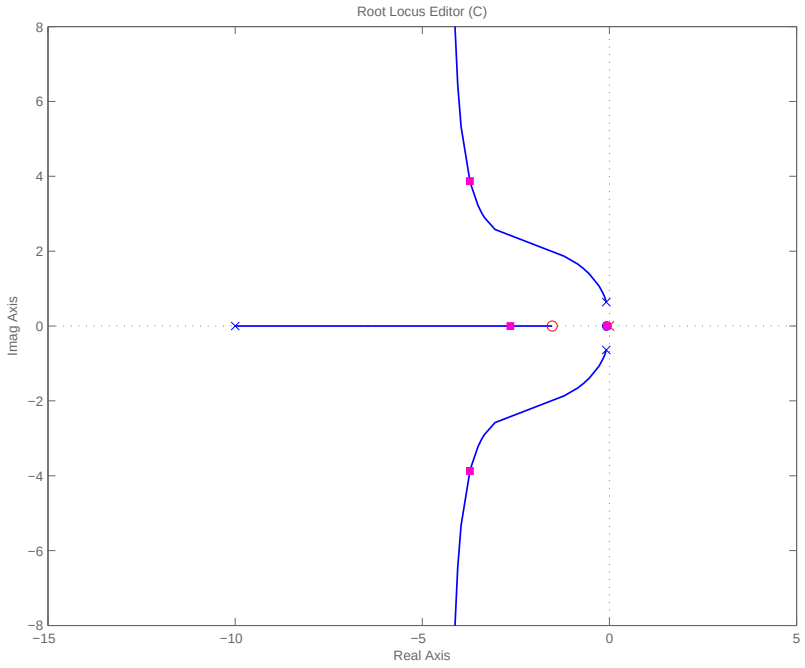


Figure 100: Root locus plot of the closed-loop system incorporating the PI speed controller. The closed-loop poles and zeros are shown, and the squares represent the closed-loop poles at $k_{V_T} = 0.773$

pushes the conjugate poles around the origin to $s = -1.92 \pm 2.13i$ to yield a damping ratio of 0.671. Figure 101 shows the step response of closed-loop system with the designed PI controller. The step response of the closed-loop system with the closed-loop zero removal PI implementation is also shown in Fig. 101, as the overshoot performance is improved at the cost of reduced rise time.

D.3 Lateral controllers design

In this section, we present a detail design of inner loop controllers for lateral-directional dynamic motion of the UAV. The roll and yaw dynamics are not, in general, decoupled from each other, the augmented control system design in the sequel should be incorporated with multivariable system analysis, for two control inputs of aileron and rudder, and two or more lateral outputs.

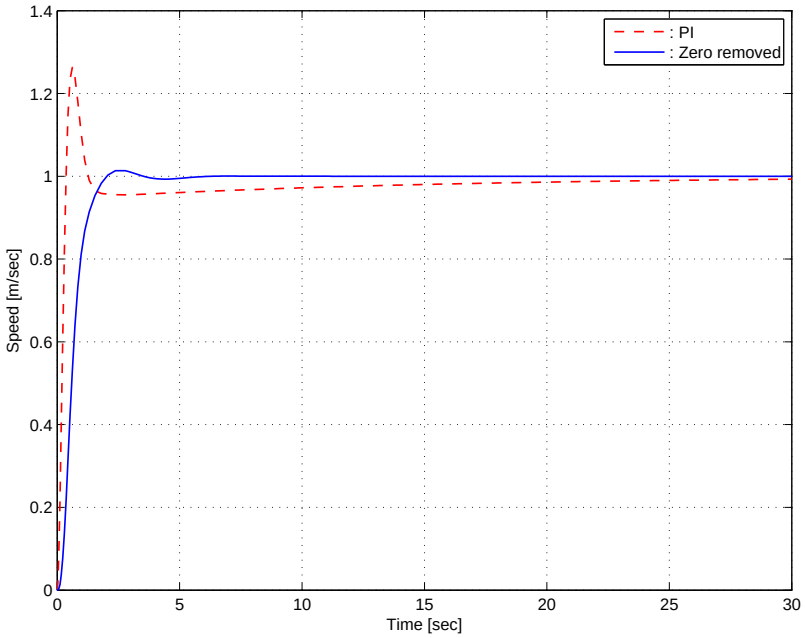


Figure 101: Step response of the closed-loop system by a unit speed reference command.

D.3.1 Yaw damper

The purpose of the yaw damper is to use the rudder to generate a yawing moment that opposes any yaw rate that builds up from the dutch roll mode. In a coordinated steady-state turn where the yaw rate is non-zero constant, the use of the yaw damper is found to be contradicting to the turn maneuver. A washout filter is used to compromise the tendency of turning and the yaw rate damping, by which the yaw rate signal due to the coordinated turn is differentiated and vanishes during steady-state turning condition. In contrast, the yaw rate signal due to the dutch-roll dynamics, which is assumed to be relatively high-frequency, is utilized in the yaw damper to suppress the dutch-roll mode. The wash out filter is given in the form of a first order high pass filter as follow,

$$G_{\mathrm{w}}(s) = \frac{\Delta r_{\mathrm{w}}}{\Delta r} = \frac{\tau_{\mathrm{w}} s}{1 + \tau_{\mathrm{w}} s}, \tag{209}$$

where the time constant τ_{w} is determined by $\tau \mathrm{w} = 1.0$ by taking into account the natural frequency of the inherent dutch-roll mode of the UAV.

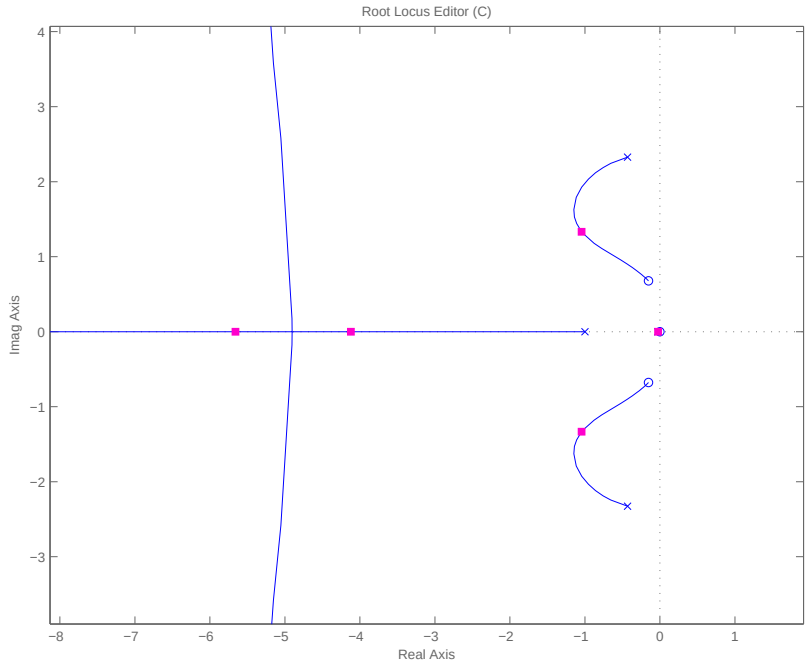


Figure 102: Root locus plot of the closed-loop system with the designed yaw washout damper. The squares represent the closed-loop poles at $k_r = 0.27$.

Figure 102 shows the root locus plot of the closed-loop system with the yaw washout damper in the feedback path. The washout gain is chosen $k_r = 0.27$, hence the dominant dutch-roll poles end up with $s = -1.0 \pm 1.55i$ with the damping ratio of 0.543. Figure 103 shows the performance of the yaw damper in conjunction with the washout filter. The aileron doublet command is used to excite the dutch-roll motion, and the yaw damper suppresses the residual roll rate effectively as well as the residual yaw rate, from the roll and yaw dutch roll coupling.

D.3.2 Roll controllers

The roll controllers consist of the roll rate damper and the roll angle controller. The roll-damping loop is less critical and is closed first. An open loop transfer function from the aileron command input (Δu_a) to the roll rate output (Δp) is obtained from using Eqs. (193) and (194a) as follows,

$$\frac{\Delta p}{\Delta u_a} = \frac{1822.4(s - 0.027)(s + 0.449 \pm 2.131i)}{(s + 0.064)(s + 0.364 \pm 2.465i)(s + 10)(s + 16.926)}, \tag{210}$$

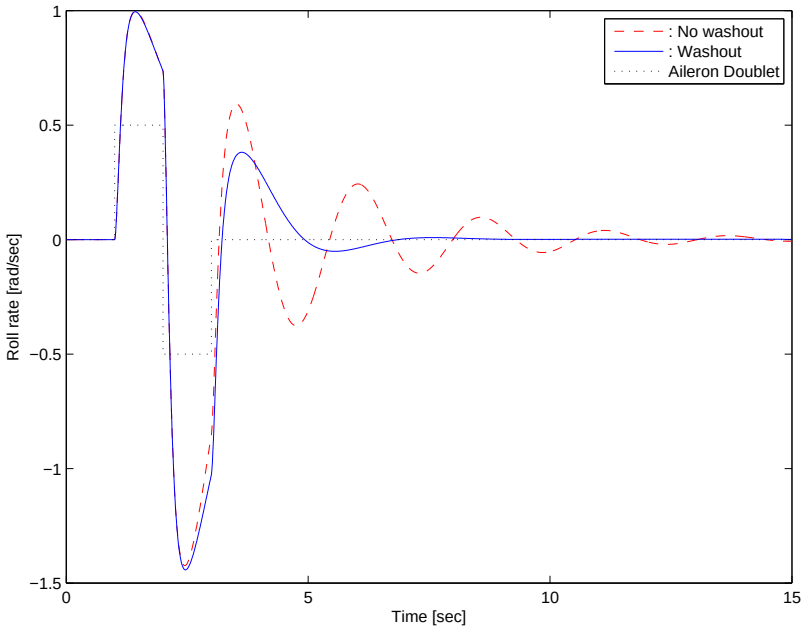


Figure 103: The effectiveness of the yaw damper to suppress the dutch roll mode oscillation.

which reveals that the plant has a non-minimum phase zero, thus excessive gain will lead to a unstable closed-loop pole. Figure 104 shows the root locus plot of the unity feedback with positive roll damping gain k_p, which indicates that a large gain will push the spiral pole to the right half plane to end up with a unstable pole. Consequently, the gain of the roll damper is chosen $k_p = 0.075$ to make the closed-loop system stable.

The closed-loop transfer function from the commanded roll rate (Δp_r) to the roll rate (Δp) after closing the roll damper is calculated as follows,

$$\frac{\Delta p}{\Delta p_r} = \frac{1822.4(s - 0.027)(s + 0.93 \pm 0.97i)(s + 5.02 \pm 1.13i)}{(s + 0.048)(s + 1.0 \pm 1.55i)(s + 4.77 \pm 1.05i)(s + 10)(s + 17.12)}. \tag{211}$$

With the designed roll damper, we design a roll angle controller that forms an outer loop of the system given in Eq. (211). In general, the roll angle is not measured directly from the inertial sensors but estimated from the attitude Kalman filter as developed in Appendix B. Since the filter introduces the time latency, the delay effect of the filter should be taken into consideration in designing the roll angle controller.

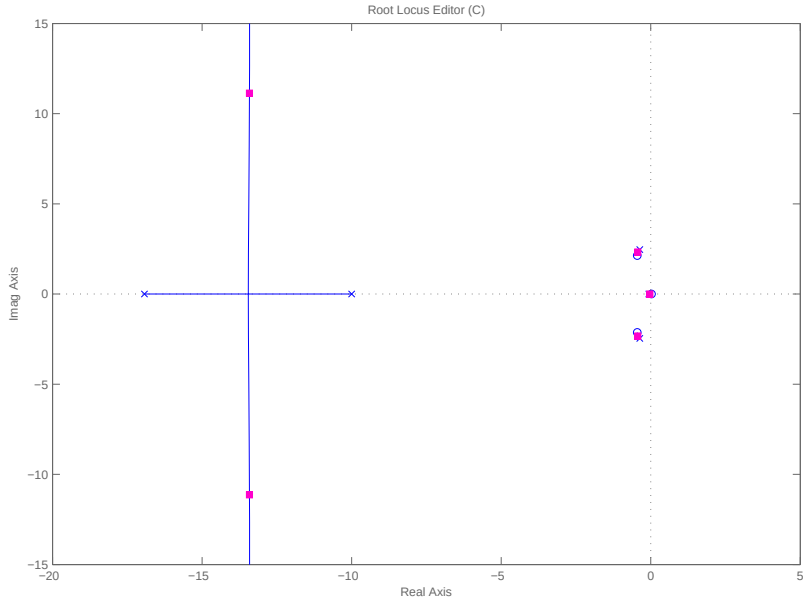


Figure 104: Root locus plot of the closed-loop system with the roll damper. The squares represent the closed-loop poles at $k_p = 0.075$.

Subsequently, the filter is modelled by a first-order low-pass filter from the actual bank angle ($\Delta\phi$) to the estimated bank angle ($\Delta\widehat{\phi}$) as follows,

$$G_f(s) = \frac{\Delta\widehat{\phi}(s)}{\Delta\phi(s)} = \frac{1}{\tau_f s + 1}, \tag{212}$$

where the time constant τ_f is chosen as $\tau_f = 0.5$ which corresponds the 0.5 [sec] time latency identified a priori.

After closing the roll damping loop, the open loop plant from the roll rate command input (Δp_r) to the estimated roll angle ($\Delta\widehat{\phi}$) is obtained as follows,

$$\frac{\Delta\widehat{\phi}}{\Delta p_r} = \frac{3655.6(s + 0.91 \pm 0.98i)(s + 5.02 \pm 1.06i)}{(s + 0.048)(s + 1.00 \pm 1.55i)(s + 2)(s + 4.77 \pm 1.05i)(s + 10)(s + 17.12)}. \tag{213}$$

Because the plant has no integrator (type 0), a PI controller is designed to remove the steady state error as follows,

$$G_\phi(s) = k_\phi \frac{s + 0.05}{s}. \tag{214}$$

Figure 105 shows the root locus plot of the closed-loop system with the roll angle PI controller. Because the closed-loop system has two dominant modes around the

origin, the gain k_ϕ is carefully chosen $k_\phi = 0.128$, after evaluating the time response of the closed-loop system. Figure 106 shows the step response of the roll angle closed-loop system. In addition, the Bode plot of the closed-loop system is displayed in Fig. 107, which shows that the closed-loop system has the gain margin of 10.6 [dB] and the phase margin of 63.3 [deg].

Figure 108 shows the entire block diagram of the closed-loop system for lateral controllers.

D.4 Discrete implementation

For the discrete implementation, the control sampling rate is first determined by 20 [Hz] taking into account the computational throughput of the RCM-3400 micro-controller. Note that the dynamics of the UAV is much slower than 20 [Hz], we assume that at this sampling rate the most dynamic characteristics are captured by the on-board sensors. The discrete implementation of the controller is computed by incorporating the bilinear mapping (Tustin's method) as follows,

$$s = \frac{2}{T}\frac{z-1}{z+1}, \tag{215}$$

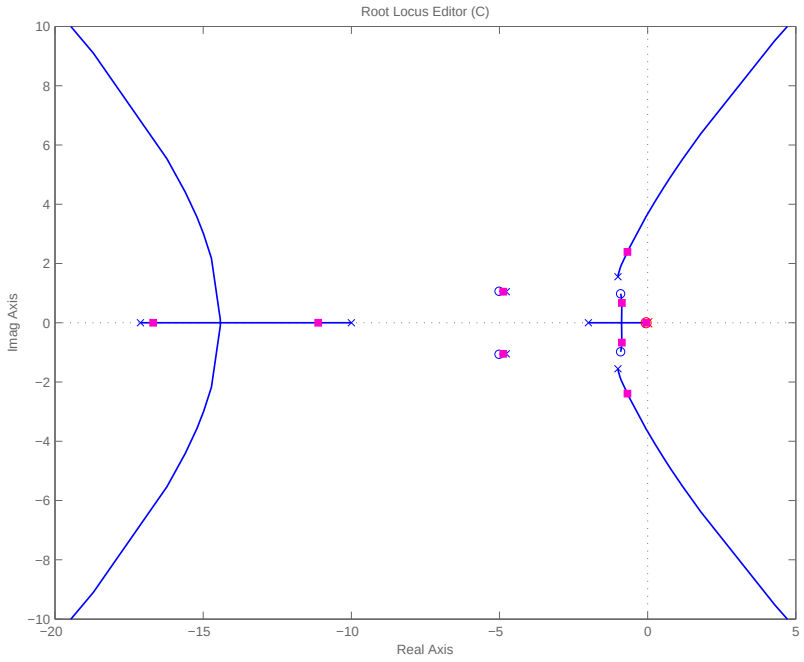
where $T = 0.05$ is used for the 20Hz sampling frequency.

The discrete version of the controller, as an example by the roll angle PI controller, can be obtained as follows,

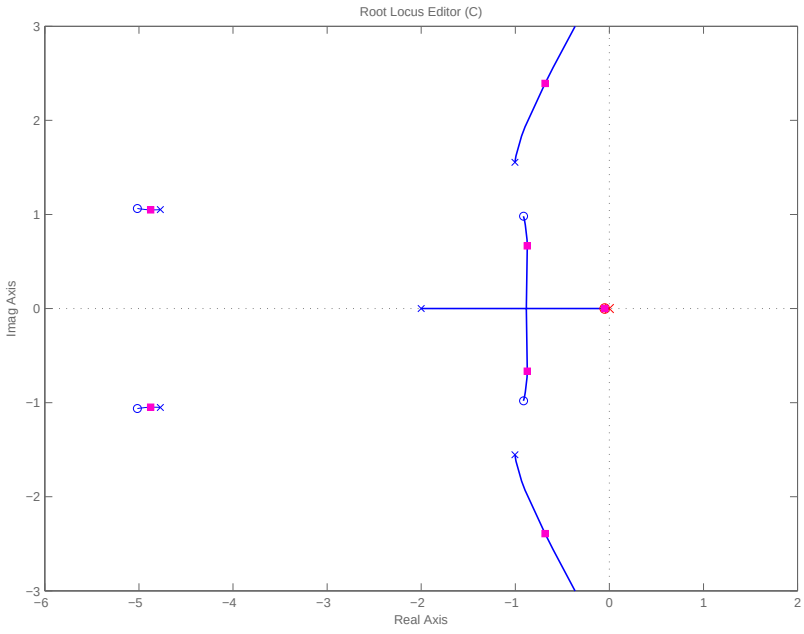
$$\begin{aligned}\frac{\Delta p_r}{\Delta \phi_e} &= k_\phi \frac{s+T_I}{s}\bigg|_{s=40\frac{z-1}{z+1}} \\ &= \frac{40k_\phi(z-1)+k_\phi T_I(z+1)}{40(z-1)} \\ &= \frac{(k_\phi+0.025k_\phi T_I)+(-k_\phi+0.025k_\phi T_I)z^{-1}}{1-z^{-1}}.\end{aligned} \tag{216}$$

Consequently, the control at time step k is obtained from the inverse z-transform,

$$\Delta p_r[k] = \Delta p_r[k-1] + (k_\phi + 0.025k_\phi T_I)\Delta\phi_e[k] + (-k_\phi + 0.025k_\phi.T_I)\Delta\phi_e[k-1] \tag{217}$$



(a) Entire display of the root locus plot of the roll angle closed-loop system.



(b) Magnified display of the root locus plot showing two dominant modes around the origin.

Figure 105: Root locus plot of the closed-loop system with the PI roll angle controller as k_ϕ varies. The squares represent the closed-loop poles at $k_\phi = 0.128$.

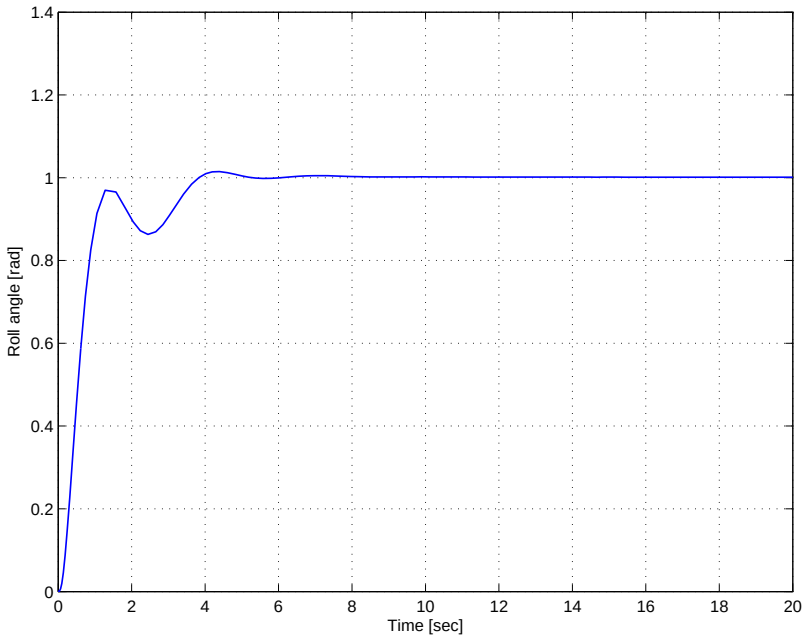


Figure 106: Step response of the closed-loop system by a unit roll angle reference command.

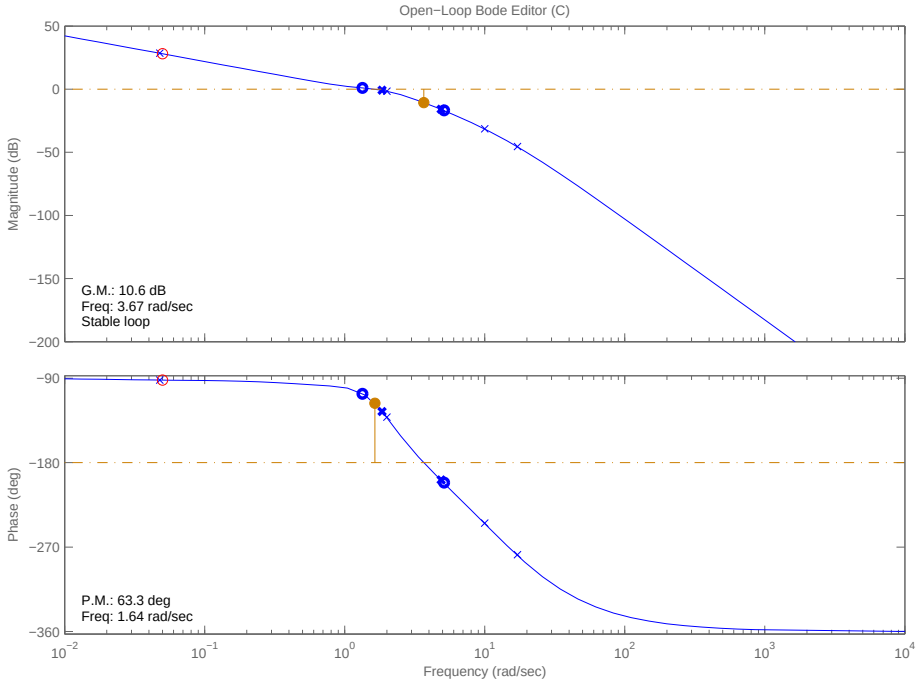


Figure 107: Bode plot of the closed-loop system with the roll angle PI controller.

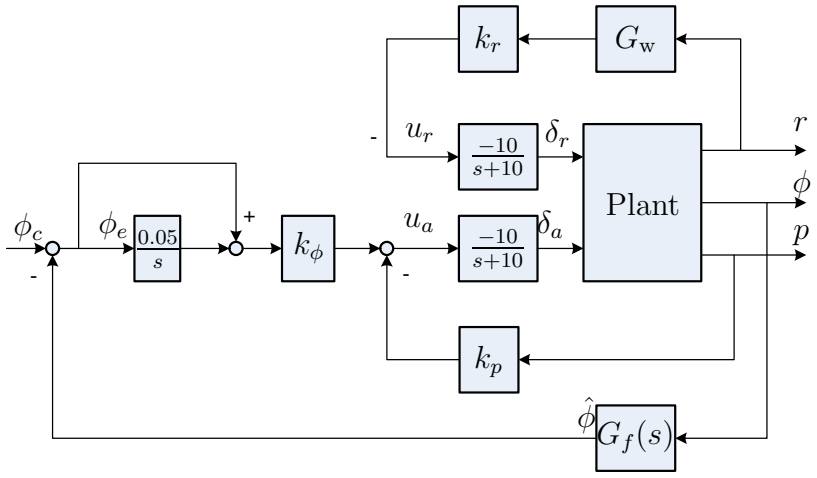


Figure 108: Entire block diagram for the lateral controllers

REFERENCES

[1] "FlightGear Flight Simulator," Oct. 2007. http://www.flightgear.org/.

[2] Adkins, C. N. and Liebeck, R. H., "Design of Optimum Propellers," *Journal of Propulsion and Power*, vol. 10, no. 5, pp. 676–682, 1994.

[3] Aguiar, A. P. and Pascoal, A. M., "Way-point Tracking of Underactuated AUVs in the Presence of Ocean Currents," in *Proceedings of the 10th Mediterranean Conference on Control and Automation*, (Lisbon, Portugal), July 2002.

[4] Allgöwer, F. and Zheng, A., *Nonlinear Model Predictive Control*, vol. 26 of *Progress in Systems and Control Theory*. Basel, Germany: Birkhäuser Verlag, 2000.

[5] Anderson, E. P. and Beard, R. W., "An Algorithmic Implementation of Constrained Extremal Control for UAVs," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, (Monterey, Canada), Aug. 2002. AIAA 2002-4470.

[6] Anderson, E. P., Beard, R. W., and McLain, T. W., "Real-Time Dynamic Trajectory Smoothing for Unmanned Air Vehicles," *IEEE Transactions on Control Systems Technology*, vol. 13, pp. 471–477, May 2005.

[7] Arinaga, S., Nakajima, S., Okabe, H., Ono, A., and Kanayama, Y., "A Motion Planning Method for an AUV," in *Proceedings of the 1996 Symposium on Autonomous Underwater Vehicle Technology*, (Monterey, CA), pp. 477–484, June 1996.

[8] Bak, M., Larsen, T., Norgaard, M., Andersen, N., Poulsen, N., and Ravn, O., "Location Estimation using Delayed Measurements," in *Proceedings of the 5th International Workshop on Advanced Motion Control (AMC98)*, pp. 180–185, June-July 1998.

[9] Barraquand, J., Langlois, B., and Latombe, J.-C., "Numerical Potential Field Techniques for Robot Path Planning," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, pp. 224–240, Mar.-Apr. 1992.

[10] Beard, R. W., McLain, T. W., Goodrich, M., and Anderson, E. P., "Coordinated Target Assignment and Intercept for Unmanned Air Vehicles," *IEEE Transactions on Robotics and Automation*, vol. 18, pp. 911–922, Dec. 2002.

[11] Behnke, S., "Local Multiresolution Path Planning," in *RoboCup 2003: Robot Soccer World Cup VII*, vol. 3020 of *Lecture Notes in Computer Science*, pp. 332–343, Berlin: Springer, 2004.

[12] Bellingham, J., Kuwata, Y., and How, J., "Stable Receding Horizon Trajectory Control for Complex Environments," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, (Austin, TX), Aug. 2003. AIAA 2003-5635.

[13] Bellingham, J., Richards, A., and How, J. P., "Receding Horizon Control of Autonomous Aerial Vehicles," in *Proceedings of the American Control Conference*, (Anchorage, AK), pp. 3741–3746, May 2002.

[14] Bemporad, A., Luca, A. D., and Oriolo, G., "Local Incremental Planning for a Car-Like Robot Navigating among Obstacles," in *Proceedings of IEEE International Conference on Robotics and Automation*, (Minneapolis, MN), pp. 1205–1211, Apr. 1996.

[15] Berglund, T., Jonsson, H., and Söderkvist, I., "An Obstacle-avoiding Minimum Variation B-spline Problem," in *Proceedings of International Conference on Geometric Modeling and Graphics*, pp. 156–161, July 2003.

[16] Blake, W. B., "Prediction of Fighter Aircraft Dynamic Derivatives using Digital Datcom," in *Third Applied Aerodynamics Conference*, (Colorado Springs, CO), Oct. 1985. AIAA-1985-4070.

[17] Boissonnat, J. D., Cérézo, A., and Leblond, J., "Shortest Paths of Bounded Curvature in the Plane," in *Proceedings of 9^{th} IEEE International Conference on Robotics and Automation*, 1992.

[18] Bortoff, S. A., "Path planning for UAVs," in *Proceedings of the American Control Conference*, (Chicago, IL), pp. 364–368, 2000.

[19] Brooks, R. A., "Solving the Find-Path Problem by Good Representation of Free Space," *IEEE Transactions on System, Man, and Cybernetics*, vol. 13, no. 3, pp. 190–193, 1983.

[20] Brown, R. G. and Hwang, P. Y. C., *Introduction to Random Signals and Applied Kalman Filtering with Matlab Exercises and Solutions*. New York, NY: John Wiley & Sons, 3rd ed., 1997.

[21] Bui, X.-N. and Souères, P., "Shortest Path Synthesis for Dubins Nonholonomic Robot," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 1, (San Diego, CA), pp. 2–7, May 1994.

[22] Burrus, C. S., Gopinath, R. A., and Guo, H., *Introduction to Wavelets and Wavelet Transforms*. Upper Saddle River, New Jersey: Prentice Hall, 1998.

[23] Calderbank, A. R., Daubechies, I., Sweldens, W., and Yeo, B.-L., "Wavelet Transforms That Map Integers to Integers," *Applied and Computational Harmonic Analysis*, vol. 5, no. 3, pp. 332–369, 1998.

[24] Camacho, E. F. and Bordons, C., *Model Predictive Control*. London, UK: Springer-Verlag, 1999.

[25] Carmo, M. D., *Differential Geometry of Curves and Surfaces*. Prentice Hall, 1976.

[26] Caruso, M. J., "Application of Magnetic Sensors for Low Cost Compass Systems." Honeywell technical article. http://www.ssec.honeywell.com/magnetic/.

[27] Caruso, M. J., "Applications of Magnetoresistive Sensors in Navigation Systems." Honeywell technical article. http://www.ssec.honeywell.com/magnetic/.

[28] Chandler, P. and Pachter, M., "Research Issues in Autonomous Control of Tactical UAVs," in *Proceedings of the American Control Conference*, (Philadelphia, PA), pp. 394–398, 1998.

[29] Chandler, P. R., Rasmussen, S., and Pachter, M., "UAV Cooperative Path Planning," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, (Denver, CO), Aug. 2000. AIAA-2000-4370.

[30] Chen, D. Z., Szczerba, R. J., and Uhran, J. J., "A Framed-Quadtree Approach for Determining Euclidean Shortest Paths in a 2-D Environment," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 5, pp. 668–681, 1997.

[31] Colgren, R. D. and Martin, K. E., "Flight Test Validation of Sideslip Estimation using Inertial Accelerations," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, (Denver, CO), Aug. 2000. AIAA-2000-4448.

[32] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C., *Introduction to Algorithms*. The MIT Press, 2nd ed., Sept. 2001.

[33] Daubechies, I. and Sweldens, W., "Factoring Wavelet Transform into Lifting Steps," *Journal of Fourier Analysis and Applications*, vol. 4, no. 3, pp. 247–269, 1998.

[34] de Boor, C., *A Practical Guide to Splines*, vol. 27 of *Applied mathematical sciences*. New York, NY: Springer-Verlag, 1978.

[35] Dijkstra, E., "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[36] Dimock, G. A., Deters, R. W., and Selig, M. S., "Icing Scenarios with the Icing Encounter Flight Simulator," in *the AIAA 41st Aerospace Sciences Meeting and Exhibit*, (Reno, NV), Jan. 2003. AIAA-2003-23.

[37] Donoho, D. L., "Smooth Wavelet Decompositions with Blocky Coefficient Kernels," in *Recent Advances in Wavelet Analysis*, pp. 1–43, Academic Press, 1993.

[38] Dubins, L. E., "On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents," *American Journal of Mathematics*, vol. 79, pp. 497–516, 1957.

[39] Dyllong, E. and Visioli, A., "Planning and Real-time Modifications of a Trajectory Using Spline Techniques," *Robotica*, vol. 21, pp. 475–482, 2003.

[40] Encarnacao, P. and Pascoal, A., "Combined Trajectory Tracking and Path Following: An Application to the Coordinated Control of Autonomous Marine Craft," in *Proceedings of the 40th IEEE Conference on Decision and Control*, (Orlando, FL), pp. 964–969, Dec. 2001.

[41] Etkin, B. and Reid, L. D., *Dynamics of Flight: Stability and Control*. New York, NY: John Wiley and Sons, 3rd ed., 1996.

[42] Finch, S. R., *Mathematical Constants*, ch. 5, pp. 331–339. Cambridge, England: Cambridge University Press, 2003.

[43] Fuller, J., Seto, D., and Meisner, R., "Optimization-Based Control for Flight Vehicles," in *AIAA Guidance, Navigation, and Control Conference*, (Denver, CO), Aug. 2000. AIAA 2000-4055.

[44] Galbraith, B., "DATCOM Predicted Aerodynamic Model," July 2004. http://www.holycows.net.

[45] Garcia, C. E., Prett, D. M., and Morari, M., "Model Predictive Control: Theory and Practice – A Survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.

[46] Geiger, B. R., Horn, J. F., DeLullo, A. M., Long, L. N., and Niessner, A. F., "Optimal Path Planning of UAVs Using Direct Collocation with Nonlinear Programming," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, (Keystone, CO), Aug. 2006. AIAA 2006-6199.

[47] Godbole, D., Samad, T., and Gopal, V., "Active Multi-Model Control for Dynamic Maneuver Optimization of Unmanned Air Vehicles," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1257–1262, 2000.

[48] Greville, T. N. E., *Theory and Applications of Spline Functions*. New York, NY: Academinc press, 1968.

[49] Hart, P. E., Nilsson, N. J., and Raphael, B., "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, pp. 100–107, 1968.

[50] Heer, V. K. and Reinfelder, H.-E., "A Comparison of Reversible Methods for Data Compression," in *Medical Imaging IV: Image Processing*, vol. Proc. SPIE Vol. 1233, pp. 354–365, 1990.

[51] Heller, M., Myschik, S., Holzapfel, F., and Sachs, G., "Low-cost Approach based on Navigation Data for Determining Angles of Attack and Sideslip for Small Aircraft," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, (Austin, TX), Aug. 2003. AIAA-2003-5777.

[52] Hwang, J. Y., Kim, J. S., Lim, S. S., and Park, K. H., "A Fast Path Planning by Path Graph Optimization," *IEEE Transactions on Systems, Man, and Cybernetics–Part A: Systems and Humans*, vol. 33, no. 1, pp. 121–128, 2003.

[53] Inman, D. J., *Engineering Vibration*. Englewood Cliffs, New Jersey: Prentice Hall, 1996.

[54] Jackins, C. L. and Tanimoto, S. L., "Oct-tree and Their Use in Representing Three Dimensional Objects," *Computer Graphics and Information Processing*, vol. 14, no. 3, pp. 249–270, 1980.

[55] Jia, D. and Vagners, J., "Parallel Evolutionary Algorithms for UAV Path Planning," in *AIAA 1^{st} Intelligent Systems Technical Conference*, (Chicago, IL), Sept. 2004. AIAA 2004-6230.

[56] Judd, K. B. and McLain, T. W., "Spline Based Path Planning for Unmanned Air Vehicles," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, (Montreal, Canada), Aug. 2001. AIAA 2001-4238.

[57] Jung, D. and Tsiotras, P., "A 3-DoF Experimental Test-Bed for Integrated Attitude Dynamics and Control Research," in *AIAA Guidance, Navigation, and Control Conference*, (Austin, Texas), 2003. AIAA 2003-5331.

[58] Jung, D. and Tsiotras, P., "Modelling and Hardware-in-the-loop Simulation for a Small Unmanned Aerial Vehicle," in *AIAA Infotech at Aerospace*, (Rohnert Park, CA), May 2007. AIAA Paper 07-2763.

[59] Jung, D. and Tsiotras, P., "Multiresolution On-Line Path Planning for Small Unmanned Aerial Vehicles," in *American Control Conference*, 2008. submitted.

[60] Kambhampati, S. and Davis, L. S., "Multiresolution Path Planning for Mobile Robots," *IEEE Journal of Robotics and Automation*, vol. 2, pp. 135–145, Sept. 1986.

[61] Kanayama, Y. and Hartman, B. I., "Smooth Local Path Planning for Autonomous Vehicles," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 3, pp. 1265–1270, May 1989.

[62] Kang, Y. and Hedrick, J. K., "Design of Nonlinear Model Predictive Controller for a Small Fixed-Wing Unmanned Aerial Vehicle," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, (Keystone, CO), Aug. 2006. AIAA-2006-6685.

[63] Karata, T. and Bullo, F., "Randomized Searches and Nonlinear Programming in Trajectory Planning," in *Proceedings of the 40th IEEE Conference on Decision and Control*, (Orlando, FL), pp. 5032–5037, Dec. 1994.

[64] Kavraki, L. and Latombe, J.-C., "Randomized Preprocessing of Configuration for Fast Path Planning," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 3, (San Diego, CA), pp. 2138–2145, May 1994.

[65] Khatib, O., "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," *International Journal of Robotics Research*, vol. 5, no. 1, pp. 396–404, 1986.

[66] Kim, B. and Tsiotras, P., "Time-Invariant Stabilization of a Unicycle-Type Mobile Robot: Theory and Experiments," in *IEEE Conference on Control Applications*, (Ancorage, AL), pp. 443–448, Sept. 2000.

[67] Kim, H. J., Shim, D. H., and Sastry, S., "Nonlinear Model Predictive Tracking Control for Rotorcraft-based Unmanned Aerial Vehicles," in *Proceedings of the American Control Conference*, (Anchorage, AK), pp. 3576–3581, May 2002.

[68] Kim, J., Pearce, R. A., and Amato, N. M., "Extracting Optimal Paths from Roadmaps for Motion Planning," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, pp. 2424–2429, Sept. 2003.

[69] Klein, R., *Concrete and Abstract Voronoi Diagrams*, vol. 400 of *Lecture notes in Computer Science*. Berlin, Germany: Springer-Verlag, 1989.

[70] Koenig, S. and Likhachev, M., "Incremental $\mathcal{A}^*$," in *Advances in Neural Information Processing Systems*, pp. 1539–1546, 2002.

[71] Koenig, S. and Likhachev, M., "$\mathcal{D}^*$ Lite," in *Proceedings of the National Conference of Artificial Intelligence*, pp. 476–483, 2002.

[72] Korf, R. E., "Real-Time Heuristic Search," *Artificial Intelligence*, vol. 42, pp. 189–211, 1990.

[73] Kuwata, Y. and How, J. P., "Stable Trajectory Design for Highly Constrained Environments using Receding Horizon Control," in *Proceedings of the 2004 American Control Conference*, (Boston, MA), pp. 902–907, June-July 2004.

[74] Lapierre, L., Soetanto, D., and Pascoal, A., "Nonlinear Path Following with Applications to the Control of Autonomous Underwater Vehicles," in *Proceedings of the 42nd IEEE Conference on Decision and Control*, (Maui, HI), pp. 1256–1261, Dec. 2003.

[75] Lapierre, L. and Soetanto, D., "Nonlinear Path-following Control of an AUV," *Ocean Engineering*, vol. 34, pp. 1734–1744, 2007.

[76] Lapierre, L., Zapata, R., and Lepinay, P., "Combined Path-following and Obstacle Avoidance Control of a Wheeled Robot," *The International Journal of Robotics Research*, vol. 26, pp. 361–375, Apr. 2007.

[77] Lapp, T. and Singh, L., "Model Predictive Control Based Trajectory Optimization for Nap-of-the-Earth (NOE) Flight Including Obstacle Avoidance," in *Proceedings of the 2004 American Control Conference*, (Boston, MA), pp. 891–896, June-July 2004.

[78] Larrabee, E. E., "Practical Design of Minimum Induced Loss Propellers," in *Society of Automotive Engineers, Business Aircraft Meeting and Exposition*, (Wichita, KS), Apr. 1979.

[79] Larsen, T., Andersen, N., Ravn, O., and Poulsen, N., "Incorporation of Time Delayed Measurements in a Discrete-time Kalman Filter," in *Proceedings of the 37th IEEE Conference on Decision and Control*, vol. 4, pp. 3972–3977, Dec. 1998.

[80] Larson, R. A., Pachter, M., and Mears, M. J., "Path Planning by Unmanned Air Vehicles for Engaging an Integrated Radar Network," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, (San Francisco, CA), Aug. 2005. AIAA 2005-6191.

[81] Latombe, J.-C., *Robot Motion Planning*. Boston, MA: Kluwer Academic Publishers, 1991.

[82] Leonardo Daga, "RS232 Blockset for Simulink," Dec. 2004. http://digilander.libero.it/LeoDaga/Simulink/RS232Blockset.htm.

[83] Leonardo Daga, "RT Blockset for Simulnk," Dec. 2004. http://digilander.libero.it/LeoDaga/Simulink/RTBlockset.htm.

[84] Lozano-Perez, T. and Wesley, M. A., "An Algorithm for Planning Collision-Free Path Among Polyhedral Obstacles," *Communications of the ACM*, vol. 22, no. 10, pp. 560–570, 1979.

[85] Luca, A. D. and Oriolo, G., "Local Incremental Planning for Nonholonomic Mobile Robots," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 1, (San Diego, CA), pp. 104–110, May 1994.

[86] Lumelsky, V. and Stepanov, A., "Dynamic Path Planning for a Mobile Automaton with Limited Information on the Environment," *IEEE Transactions on Automatic Control*, vol. 31, pp. 1058–1063, Nov. 1986.

[87] Lutterkort, D. and Peters, J., "Smooth Paths in a Polygonal Channel," in *Proceedings of the fifteenth Annual Symposium on Computational Geometry*, (Miami Beach, FL), pp. 316–321, 1999.

[88] Lutterkort, D. and Peters, J., "Tight Linear Envelopes for Splines ," *Numerische Mathematik*, vol. 89, pp. 735–748, Oct. 2001.

[89] Madras, N. and Slade, G., *The Self-Avoiding Walk*. Boston, MA: Birkhäuser, 1993.

[90] Mallat, S. G., "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 2, pp. 674–693, July 1989.

[91] Marti, K. and Qu, S., "Path Planning for Robots by Stochastic Optimization Methods," *International Journal of Intelligent and Robotic Systems*, vol. 22, pp. 117–127, June 1998.

[92] Martin Hepperle, "JavaProp - Design and Analysis of Propellers." http://www.mh-aerotools.de/airfoils/javaprop.htm.

[93] McLain, T., Chandler, P., and Pachter, M., "A Decomposition Strategy for Optimal Coordination of Unmanned Air Vehicles," in *Proceedings of the American Control Conference*, (Chicago, IL), pp. 369–373, 2000.

[94] McLain, T. W. and Beard, R. W., "Coordination Variables, Coordination Functions, and Cooperative Timing Missions," *Journal of Guidance, Control, and Dynamics*, vol. 28, no. 1, pp. 150–161, 2005.

[95] Mertz, P. and Gray, F., "A Theory of Scanning and Its Relation to the Characteristics of the Transmitted Signal in Telephotography and Television," *Bell System Technical Journal*, vol. 13, pp. 464–515, 1934.

[96] Micaelli, A. and Samson, C., "Trajectory Tracking for Unicycle-type and Two-Steering-Wheels Mobile Robots," Tech. Rep. 2097, INRIA, Sophia-Antipolis, Nov. 1993.

[97] Morari, M. and Lee, J., "Model Predictive Control: Past, Present, and Future," *Computers and Chemical Engineering*, vol. 23, no. 4, pp. 667–682, 1999.

[98] Motorola, Inc., *Motorola GPS Products - Oncore User's Guide*, Aug. 2002. Revision 5.0.

[99] Nairn, D., Peters, J., and Lutterkort, D., "Sharp, Quantitative Bounds on the Distance Between a Polynomial Piece and its Bézier Control Polygon," *Computer Aided Geometric Design*, vol. 16, pp. 613–631, 1999.

[100] Nelson, D. R., Barber, D. B., McLain, T. W., and Beard, R. W., "Vector Field Path Following for Small Unmanned Air Vehicles," in *Proceedings of the 2006 American Control Conference*, (Minneapolis, MN), pp. 5788–5794, June 2006.

[101] Neus, M. and Maouche, S., "Motion Planning using the Modified Visibility Graph," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, vol. 4, (Tokyo, Japan), pp. 651–655, Oct. 1999.

[102] Niculescu, M., "Lateral Track Control Law for Aerosonde UAV," in *39th AIAA Aerospace Sciences Meeting and Exhibit*, (Reno, NV), Jan. 2001. A01-16013.

[103] Noborio, H., Naniwa, T., and Arimoto, S., "A Quadtree-Based Path-Planning Algorithm for a Mobile Robot," *Journal of Robotic Systems*, vol. 7, no. 4, pp. 555–574, 1990.

[104] Pai, D. K. and Reissell, L.-M., "Multiresolution Rough Terrain Motion Planning," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 1, pp. 19–33, 1998.

[105] Park, S., *Avionics and Control System Development for Mid-Air Rendezvous of Two Unmanned Aerial Vehicles*. Ph.d. thesis, Massachusetts Institute of Technology, Boston, MA, Feb. 2004.

[106] Park, S., Deyst, J., and How, J. P., "A New Nonlinear Guidance Logic for Trajectory Tracking," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, (Providence, RI), Aug. 2004. AIAA-2004-4900.

[107] Perry, A. R., "The FlightGear Flight Simulator," in *USENIX Annual Technical Conference*, UseLinux SIG sessions, (Boston, MA), June-July 2004.

[108] Pettersen, K. Y. and Lefeber, E., "Way-point Tracking Control of Ships," in *Proceedings of the 40th IEEE Conference on Decision and Control*, (Orlando, FL), pp. 940–945, Dec. 2001.

[109] Piegl, L. and Tiller, W., *The NURBS Book*. Monographs in Visual Communication, Berlin Heidelberg: Springer-Verlag, 2 ed., 1997.

[110] Pirzadeh, A. and Snyder, W., "A Unified Solution to Coverage and Search in Explored and Unexplored Terrains using Indirect Control," in *IEEE International Conference on Robotics and Automation*, vol. 3, pp. 2113–2119, May 1990.

[111] Podscedkowski, L., "Path Planner for Nonholonomic Mobile Robot with Fast Replanning Procedure," in *Proceedings of the 1998 IEEE International Conference on Robotics & Automation*, (Leuven, Belgium), pp. 3588–3593, May 1998.

[112] Podsedkowski, L., Nowakowski, J., Idzikowski, M., and Visvary, I., "Modified $\mathcal{A}^*$ Algorithm Suitable for Online Car-like Mobile Robot Control," in *Proceedings of the First Workshop on Robot Motion and Control*, pp. 235–240, June 1999.

[113] Prasanth, R., Bošković, J., Li, S.-M., and Mehra, R., "Initial Study of Autonomous Trajectory Generation for Unmanned Aerial Vehicles," in *Proceedings of the 40^{th} IEEE Conference on Decision and Control*, (Orlando, FL), pp. 640–645, 2001.

[114] Raghunathan, A. U., Gopal, V., Subramanian, D., Biegler, L. T., and Samad, T., "Dynamic Optimization Strategies for Three-Dimensional Conflict Resolution of Multiple Aircraft," *Journal of Guidance, Control, and Dynamics*, vol. 27, no. 4, pp. 586–594, 2004.

[115] Rathbun, D., Kragelund, S., Pongpunwattana, A., and Capozzi, B., "An Evolution Based Path Planning Algorithm for Autonomous Motion of a UAV Through Uncertain Environments," in *Proceedings of the 21^{st} Digital Avionics Systems Conference*, vol. 2, pp. 8D2(1)–8D2(12), 2002.

[116] Ren, W. and Beard, R. W., "Trajectory Tracking for Unmanned Air Vehicles with Velocity and Heading Rate Constraints," *IEEE Transactions on Control Systems Technology*, vol. 12, pp. 706–716, Sept. 2004.

[117] Richards, A. and How, J., "Aircraft Trajectory Planning with Collision Avoidance using Mixed Integer Linear Programming," in *Proceedings of the American Control Conference*, (Anchorage, AK), pp. 1936–1940, 2002.

[118] Richards, A., How, J., Schouwenaars, T., and Feron, E., "Plume Avoidance Maneuver Planning Using Mixed Integer Linear Programming," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, (Montreal, Canada), Aug. 2001. AIAA-2001-4091.

[119] Richards, A., Kuwata, Y., and How, J., "Experimental Demonstrations of Real-time MILP Control," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, (Austin, TX), Aug. 2003. AIAA 2003-5802.

[120] Rysdyk, R., "UAV Path Following for Constant Line-Of-Sight," in *2nd AIAA Unmanned Unlimited Conference and Workshop and Exhibit*, (San Diego, CA), Sept. 2003. AIAA-2003-6626.

[121] Samad, T., Gorinevsky, D., and Stoffelen, F., "Dynamic Multiresolution Route Optimization for Autonomous Aircraft," in *Proceedings of the IEEE*

International Symposium on Intelligent Control, (Mexico City, Mexico), pp. 13–18, Sept. 2001.

[122] Scheuer, A. and Laugier, C., "Planning Sub-Optimal and Continuous-Curvature Paths for Car-Like Robots," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Victoria, B. C., Canada), pp. 25–31, Oct. 1998.

[123] Schouwenaars, T., How, J., and Feron, E., "Receding Horizon Path Planning with Implicit Safety Guarantees," in *Proceedings of the 2004 American Control Conference*, (Boston, MA), pp. 5576–5581, June-July 2004.

[124] Schouwenaars, T., Moor, B. D., Feron, E., and How, J., "Mixed Integer Programming for Multi-Vehicle Path Planning," in *Proceedings of the European Control Conference*, (Porto, Portugal), pp. 2603–2608, 2001.

[125] Schouwenaars, T., Valenti, M., Feron, E., and How, J., "Implementation and Flight Test Results of MILP-based UAV Guidance," in *2005 IEEE Conference on Aerospace*, pp. 1–13, Mar. 2005.

[126] Shanmugavel, M., Tsourdos, A., Zbikowski, R., and White, B. A., "3D Dubins Sets Based Coordinated Path Planning for Swarm of UAVs," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, (Keystone, CO), Aug. 2006. AIAA 2006-6211.

[127] Shaw, A., Barnes, D., and Summers, P., "Landmark Recognition for Localisation and Navigation of Aerial Vehicles," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, pp. 42– 47, Oct. 2003.

[128] Shim, D. H., Chung, H., Kim, H. J., and Sastry, S., "Autonomous Exploration in Unknown Urban Environments for Unmanned Aerial Vehicles," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, (San Francisco, CA), Aug. 2005. AIAA 2005-6478.

[129] Shim, D. H., Kim, H. J., and Sastry, S., "Decentralized Nonlinear Model Predictive Control of Multiple Flying Robots," in *Proceedings of the 42nd IEEE Conference on Decision and Control*, (Maui, HI), pp. 3621–3626, Dec. 2003.

[130] Shin, K. and McKay, N., "A Dynamic Programming Approach to Trajectory Planning of Robotic Manipulators," *IEEE Transactions on Automatic Control*, vol. 31, no. 6, pp. 491–500, 1986.

[131] Singh, L. and Fuller, J., "Trajectory Generation for a UAV in Urban Terrain, using Nonlinear MPC," in *Proceedings of the American Control Conference*, (Arlington, VA), pp. 2301–2308, June 2001.

[132] Smetana, F. O., *Computer Assisted Analysis of Aircraft Performance, Stability, and Control*. New York: McGraw-Hill, 1984.

[133] Sorton, E. F. and Hammaker, S., "Simulated Flight Testing of an Autonomous Unmanned Aerial Vehicle Using FlightGear," in *Infotech@Aerospace*, (Arlington, VA), Sept. 2005. AIAA 2005-7083.

[134] Spooner, J. T., Maggiore, M., Raúl Ordóñez, and Passino, K. M., *Stable Adaptive Control and Estimation for Nonlinear Systems*. New York, NY: A John Wiley & Sons, Inc., 2002.

[135] Stentz, A., "Optimal and Efficient Path Planning for Partially-Known Environments," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 4, pp. 3310–3317, May 1994.

[136] Stentz, A., "Map-based Strategies for Robot Navigation in Unknown Environments," in *Proceedings AAAI 1996 Spring Symposium on Planning with Incomplete Information for Robot Problems*, (Menlo Park, CA), pp. 110–116, 1996.

[137] Stevens, B. L. and Lewis, F. L., *Aircraft Control and Simulation*. Hobokren, NJ: John Wiley & Sons, 2nd ed., 2003.

[138] Sweldens, W. and Schröder, P., "Building Your Own Wavelets at Home," in *Wavelets in Computer Graphics*, pp. 15–87, ACM SIGGRAPH Course notes, 1996.

[139] Sweldens, W., "The Lifting Scheme: A New Philosophy in Biorthogonal Wavelet Constructions," in *Wavelet Applications in Signal and Image Processing III*, pp. 68–79, 1995.

[140] Sweldens, W., "The Lifting Scheme: A Construction of Second Generation Wavelets," *SIAM Journal on Mathematical Analysis*, vol. 29, no. 2, pp. 511–546, 1997.

[141] Thorpe, C. E., "Path Relaxation: Path Planning for a Mobile Robot," in *National Conference on Artificial Intelligence*, 1984.

[142] Tsiotras, P. and Bakolas, E., "A Hierarchical On-Line Path Planning Scheme using Wavelets," in *Proceedings of the European Control Conference*, (Kos, Greece), July 2007.

[143] Twigg, S., Calise, A., and Johnson, E., "On-line Trajectory Optimization for Autonomous Air Vehicles," in *AIAA Guidance, Navigation, and Control Conference*, (Austin, TX), pp. AIAA 2003–5522, 2003.

[144] Unmanned Dynamics, http://www.u-dynamics.com/, *AeroSim User's Guide - Aeronautical Simulation Blockset Ver. 1.2*.

[145] Uytterhoeven, G., Roose, D., and Bultheel, A., "Wavelet transforms using lifting scheme," technical report ita-wavelets report wp 1.1, Katholieke Universiteit Leuven, Belgium, Apr. 1997.

[146] Valenti, M., Schouwenaars, T., Kuwata, Y., Feron, E., and How, J., "Implementation of a Manned Vehicle - UAV Mission System," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, (Providence, RI), Aug. 2004. AIAA 2004-5142.

[147] Vázquez G., B., Sossa A., J. H., and Díaz-de-León S., J. L., "Auto Guided Vehicle Control Using Expanded Time B-splines," in *IEEE International Conference on Systems, Man, and Cybernetics*, vol. 3, (San Antonio, TX), pp. 2786–2791, Oct. 1994.

[148] Vörös, J., "Low-cost Implementation of Distance Maps for Path Planning using Matrix Quadtrees and Octrees," *Robotics and Computer Integrated Manufacturing*, vol. 17, pp. 447–459, 2001.

[149] Walnut, D. F., *An Introduction to Wavelet Analysis*. Boston: Birkhäuser, 2002.

[150] Williams, J. E. and Vukelich, S. R., "The USAF Stability and Control DATCOM," tech. rep., McDonnell Douglas Astronautics Company, St Louis, MO, 1979. AFFDL-TR-79-3032.

[151] Yang, G. and Kapila, V., "Optimal Path Planning for Unmanned Air Vehicles with Kinematic and Tactical Constraints," in *Proceedings of the 41st IEEE Conference on Decision and Control*, (Las Vegas, NV), pp. 1301–1306, Dec. 2002.

[152] Yang, H. I. and Zhao, Y. J., "Trajectory Planning for Autonomous Aerospace Vehicles amid Known Obstacles and Conflicts," *Journal of Guidance, Control, and Dynamics*, vol. 27, pp. 997–1008, Nov.-Dec. 2004.

VITA

Dongwon Jung was born in Chungju, Korea, on January 1973. He received a bachelor's and a master's degrees at the Seoul National University, Seoul, Korea in 1998 and 2000, respectively. In Fall 2001, he began his studies for a Ph.D degree in the School of Aerospace Engineering at Georgia Institute of Technology, Atlanta, Georgia, USA.

His current research interest is to develop control algorithms for embedded systems that take into account the limited hardware resources, including multiresolution path planning, path generation/tracking, and validation of control algorithms through experiments.